The Java API XML and DOM Sources

Marco Mendieta Parihuancollo

Versión 036MPACAGAm2 (2025-10-18)

Secuencia 1: Source code

```
root:           C:\Program Files\Java\jdk1.8.0_131\src.zip
-----------------------------------------------------------------------------------------------
Section                        Package (49)                        Files (513)   Lines (92909)
-----------------------------------------------------------------------------------------------
Xml                            javax.xml                           1             392
Xml Bind                       javax.xml.bind                      32            9120
Xml Bind Annotation            javax.xml.bind.annotation           35            4597
Xml Bind Annotation Adapters   javax.xml.bind.annotation.adapters   6             631
Xml Bind Attachment            javax.xml.bind.attachment           2             339
Xml Bind Helpers               javax.xml.bind.helpers              9             1863
Xml Bind Util                  javax.xml.bind.util                 4             618
Xml Crypto                     javax.xml.crypto                    14            1527
Xml Crypto Dom                 javax.xml.crypto.dom                3             383
Xml Crypto Dsig                javax.xml.crypto.dsig               17            3005
Xml Crypto Dsig Dom            javax.xml.crypto.dsig.dom           2             344
Xml Crypto Dsig Keyinfo        javax.xml.crypto.dsig.keyinfo       8             1286
Xml Crypto Dsig Spec           javax.xml.crypto.dsig.spec          10            884
Xml Datatype                   javax.xml.datatype                  7             3894
Xml Namespace                  javax.xml.namespace                 2             804
Xml Parsers                    javax.xml.parsers                   8             2570
Xml Soap                       javax.xml.soap                      28            5688
Xml Stream                     javax.xml.stream                    17            4097
Xml Stream Events              javax.xml.stream.events             14            1045
Xml Stream Util                javax.xml.stream.util               4             561
Xml Transform                  javax.xml.transform                 14            2501
Xml Transform Dom              javax.xml.transform.dom             3             548
Xml Transform Sax              javax.xml.transform.sax             5             647
Xml Transform Stax             javax.xml.transform.stax            2             419
Xml Transform Stream           javax.xml.transform.stream          2             487
Xml Validation                 javax.xml.validation                9             2869
Xml Ws                         javax.xml.ws                        31            3901
Xml Ws Handler                 javax.xml.ws.handler                6             491
Xml Ws Handler Soap            javax.xml.ws.handler.soap           2             148
Xml Ws Http                    javax.xml.ws.http                   2             98
Xml Ws Soap                    javax.xml.ws.soap                   6             794
Xml Ws Spi                     javax.xml.ws.spi                    5             1536
Xml Ws Spi Http                javax.xml.ws.spi.http               4             576
Xml Ws Wsaddressing            javax.xml.ws.wsaddressing           3             548
Xml Xpath                      javax.xml.xpath                     13            2172
Dom                            org.w3c.dom                         28            5593
```

```
Dom Bootstrap              org.w3c.dom.bootstrap          1          429
Dom Css                    org.w3c.dom.css                22         3821
Dom Events                 org.w3c.dom.events             8          930
Dom Html                   org.w3c.dom.html               56         5174
Dom Ls                     org.w3c.dom.ls                 11         2113
Dom Ranges                 org.w3c.dom.ranges             3          575
Dom Stylesheets            org.w3c.dom.stylesheets        5          440
Dom Traversal              org.w3c.dom.traversal          4          641
Dom Views                  org.w3c.dom.views              2          115
Dom Xpath                  org.w3c.dom.xpath              6          724
Sax                        org.xml.sax                    17         3966
Sax Ext                    org.xml.sax.ext                8          1493
Sax Helpers                org.xml.sax.helpers            12         5512
-------------------------------------------------------------------------------------------
Líneas de código: 92909 (1786 páginas)
```

Contents

1 Xml **2**
1.1 XMLConstants.java 2

2 Xml Bind **10**
2.1 Binder.java 10
2.2 ContextFinder.java 18
2.3 DataBindingException.java 30
2.4 DatatypeConverter.java 31
2.5 DatatypeConverterImpl.java 45
2.6 DatatypeConverterInterface.java 65
2.7 Element.java 75
2.8 GetPropertyAction.java 76
2.9 JAXB.java 77
2.10 JAXBContext.java 89
2.11 JAXBElement.java 104
2.12 JAXBException.java 108
2.13 JAXBIntrospector.java 112
2.14 JAXBPermission.java 114
2.15 MarshalException.java 115
2.16 Marshaller.java 117
2.17 Messages.java 133
2.18 NotIdentifiableEvent.java 135
2.19 ParseConversionEvent.java 136
2.20 PrintConversionEvent.java 137
2.21 PropertyException.java 138
2.22 SchemaOutputResolver.java 140
2.23 TypeConstraintException.java 142
2.24 UnmarshalException.java 145
2.25 Unmarshaller.java 147
2.26 UnmarshallerHandler.java 169
2.27 ValidationEvent.java 171
2.28 ValidationEventHandler.java 173
2.29 ValidationEventLocator.java 175
2.30 ValidationException.java 176

2.31 Validator.java . . . 178
2.32 WhiteSpaceProcessor.java . . . 184

3 Xml Bind Annotation 188
3.1 DomHandler.java . . . 188
3.2 W3CDomHandler.java . . . 190
3.3 XmlAccessOrder.java . . . 192
3.4 XmlAccessType.java . . . 193
3.5 XmlAccessorOrder.java . . . 194
3.6 XmlAccessorType.java . . . 196
3.7 XmlAnyAttribute.java . . . 198
3.8 XmlAnyElement.java . . . 199
3.9 XmlAttachmentRef.java . . . 205
3.10 XmlAttribute.java . . . 207
3.11 XmlElement.java . . . 209
3.12 XmlElementDecl.java . . . 214
3.13 XmlElementRef.java . . . 218
3.14 XmlElementRefs.java . . . 223
3.15 XmlElementWrapper.java . . . 224
3.16 XmlElements.java . . . 227
3.17 XmlEnum.java . . . 231
3.18 XmlEnumValue.java . . . 232
3.19 XmlID.java . . . 235
3.20 XmlIDREF.java . . . 236
3.21 XmlInlineBinaryData.java . . . 241
3.22 XmlList.java . . . 242
3.23 XmlMimeType.java . . . 245
3.24 XmlMixed.java . . . 246
3.25 XmlNs.java . . . 248
3.26 XmlNsForm.java . . . 250
3.27 XmlRegistry.java . . . 251
3.28 XmlRootElement.java . . . 252
3.29 XmlSchema.java . . . 255
3.30 XmlSchemaType.java . . . 259
3.31 XmlSchemaTypes.java . . . 261
3.32 XmlSeeAlso.java . . . 263
3.33 XmlTransient.java . . . 264
3.34 XmlType.java . . . 266
3.35 XmlValue.java . . . 275

4 Xml Bind Annotation Adapters 277
4.1 CollapsedStringAdapter.java . . . 277
4.2 HexBinaryAdapter.java . . . 280
4.3 NormalizedStringAdapter.java . . . 281
4.4 XmlAdapter.java . . . 283
4.5 XmlJavaTypeAdapter.java . . . 286
4.6 XmlJavaTypeAdapters.java . . . 289

5 Xml Bind Attachment 290
5.1 AttachmentMarshaller.java . . . 290
5.2 AttachmentUnmarshaller.java . . . 294

6 Xml Bind Helpers **297**
6.1 AbstractMarshallerImpl.java . . . 297
6.2 AbstractUnmarshallerImpl.java . . . 306
6.3 DefaultValidationEventHandler.java . . . 315
6.4 Messages.java . . . 317
6.5 NotIdentifiableEventImpl.java . . . 319
6.6 ParseConversionEventImpl.java . . . 321
6.7 PrintConversionEventImpl.java . . . 323
6.8 ValidationEventImpl.java . . . 324
6.9 ValidationEventLocatorImpl.java . . . 328

7 Xml Bind Util **333**
7.1 JAXBResult.java . . . 333
7.2 JAXBSource.java . . . 336
7.3 Messages.java . . . 341
7.4 ValidationEventCollector.java . . . 343

8 Xml Crypto **345**
8.1 AlgorithmMethod.java . . . 345
8.2 Data.java . . . 346
8.3 KeySelector.java . . . 347
8.4 KeySelectorException.java . . . 350
8.5 KeySelectorResult.java . . . 353
8.6 MarshalException.java . . . 354
8.7 NoSuchMechanismException.java . . . 357
8.8 NodeSetData.java . . . 360
8.9 OctetStreamData.java . . . 361
8.10 URIDereferencer.java . . . 363
8.11 URIReference.java . . . 365
8.12 URIReferenceException.java . . . 366
8.13 XMLCryptoContext.java . . . 369
8.14 XMLStructure.java . . . 374

9 Xml Crypto Dom **375**
9.1 DOMCryptoContext.java . . . 375
9.2 DOMStructure.java . . . 379
9.3 DOMURIReference.java . . . 381

10 Xml Crypto Dsig **382**
10.1 CanonicalizationMethod.java . . . 382
10.2 DigestMethod.java . . . 384
10.3 Manifest.java . . . 386
10.4 Reference.java . . . 388
10.5 SignatureMethod.java . . . 391
10.6 SignatureProperties.java . . . 393
10.7 SignatureProperty.java . . . 395
10.8 SignedInfo.java . . . 397
10.9 Transform.java . . . 399
10.10 TransformException.java . . . 402
10.11 TransformService.java . . . 405
10.12 XMLObject.java . . . 412
10.13 XMLSignContext.java . . . 415
10.14 XMLSignature.java . . . 416

10.15 XMLSignatureException.java . . . 421
10.16 XMLSignatureFactory.java . . . 424
10.17 XMLValidateContext.java . . . 439

11 Xml Crypto Dsig Dom **440**
11.1 DOMSignContext.java . . . 440
11.2 DOMValidateContext.java . . . 444

12 Xml Crypto Dsig Keyinfo **447**
12.1 KeyInfo.java . . . 447
12.2 KeyInfoFactory.java . . . 449
12.3 KeyName.java . . . 460
12.4 KeyValue.java . . . 461
12.5 PGPData.java . . . 464
12.6 RetrievalMethod.java . . . 466
12.7 X509Data.java . . . 468
12.8 X509IssuerSerial.java . . . 470

13 Xml Crypto Dsig Spec **472**
13.1 C14NMethodParameterSpec.java . . . 472
13.2 DigestMethodParameterSpec.java . . . 473
13.3 ExcC14NParameterSpec.java . . . 474
13.4 HMACParameterSpec.java . . . 476
13.5 SignatureMethodParameterSpec.java . . . 478
13.6 TransformParameterSpec.java . . . 479
13.7 XPathFilter2ParameterSpec.java . . . 479
13.8 XPathFilterParameterSpec.java . . . 481
13.9 XPathType.java . . . 484
13.10 XSLTTransformParameterSpec.java . . . 488

14 Xml Datatype **490**
14.1 DatatypeConfigurationException.java . . . 490
14.2 DatatypeConstants.java . . . 491
14.3 DatatypeFactory.java . . . 497
14.4 Duration.java . . . 519
14.5 FactoryFinder.java . . . 538
14.6 SecuritySupport.java . . . 544
14.7 XMLGregorianCalendar.java . . . 546

15 Xml Namespace **567**
15.1 NamespaceContext.java . . . 567
15.2 QName.java . . . 573

16 Xml Parsers **582**
16.1 DocumentBuilder.java . . . 582
16.2 DocumentBuilderFactory.java . . . 589
16.3 FactoryConfigurationError.java . . . 601
16.4 FactoryFinder.java . . . 604
16.5 ParserConfigurationException.java . . . 610
16.6 SAXParser.java . . . 611
16.7 SAXParserFactory.java . . . 621
16.8 SecuritySupport.java . . . 630

17 Xml Soap **632**
17.1 AttachmentPart.java . . . 632
17.2 Detail.java . . . 642
17.3 DetailEntry.java . . . 644
17.4 FactoryFinder.java . . . 645
17.5 MessageFactory.java . . . 650
17.6 MimeHeader.java . . . 654
17.7 MimeHeaders.java . . . 655
17.8 Name.java . . . 660
17.9 Node.java . . . 662
17.10 SAAJMetaFactory.java . . . 664
17.11 SAAJResult.java . . . 667
17.12 SOAPBody.java . . . 669
17.13 SOAPBodyElement.java . . . 675
17.14 SOAPConnection.java . . . 676
17.15 SOAPConnectionFactory.java . . . 678
17.16 SOAPConstants.java . . . 680
17.17 SOAPElement.java . . . 683
17.18 SOAPElementFactory.java . . . 693
17.19 SOAPEnvelope.java . . . 696
17.20 SOAPException.java . . . 700
17.21 SOAPFactory.java . . . 703
17.22 SOAPFault.java . . . 709
17.23 SOAPFaultElement.java . . . 719
17.24 SOAPHeader.java . . . 719
17.25 SOAPHeaderElement.java . . . 724
17.26 SOAPMessage.java . . . 728
17.27 SOAPPart.java . . . 736
17.28 Text.java . . . 742

18 Xml Stream **742**
18.1 EventFilter.java . . . 742
18.2 FactoryConfigurationError.java . . . 744
18.3 FactoryFinder.java . . . 746
18.4 Location.java . . . 753
18.5 SecuritySupport.java . . . 755
18.6 StreamFilter.java . . . 757
18.7 XMLEventFactory.java . . . 758
18.8 XMLEventReader.java . . . 767
18.9 XMLEventWriter.java . . . 769
18.10 XMLInputFactory.java . . . 774
18.11 XMLOutputFactory.java . . . 784
18.12 XMLReporter.java . . . 791
18.13 XMLResolver.java . . . 792
18.14 XMLStreamConstants.java . . . 793
18.15 XMLStreamException.java . . . 796
18.16 XMLStreamReader.java . . . 798
18.17 XMLStreamWriter.java . . . 812

19 Xml Stream Events **822**
19.1 Attribute.java . . . 822
19.2 Characters.java . . . 823
19.3 Comment.java . . . 825
19.4 DTD.java . . . 826
19.5 EndDocument.java . . . 827
19.6 EndElement.java . . . 828
19.7 EntityDeclaration.java . . . 829
19.8 EntityReference.java . . . 831
19.9 Namespace.java . . . 832
19.10 NotationDeclaration.java . . . 833
19.11 ProcessingInstruction.java . . . 834
19.12 StartDocument.java . . . 835
19.13 StartElement.java . . . 837
19.14 XMLEvent.java . . . 839

20 Xml Stream Util **843**
20.1 EventReaderDelegate.java . . . 843
20.2 StreamReaderDelegate.java . . . 845
20.3 XMLEventAllocator.java . . . 851
20.4 XMLEventConsumer.java . . . 852

21 Xml Transform **854**
21.1 ErrorListener.java . . . 854
21.2 FactoryFinder.java . . . 856
21.3 OutputKeys.java . . . 863
21.4 Result.java . . . 866
21.5 SecuritySupport.java . . . 868
21.6 Source.java . . . 870
21.7 SourceLocator.java . . . 871
21.8 Templates.java . . . 873
21.9 Transformer.java . . . 875
21.10 TransformerConfigurationException.java . . . 881
21.11 TransformerException.java . . . 883
21.12 TransformerFactory.java . . . 890
21.13 TransformerFactoryConfigurationError.java . . . 899
21.14 URIResolver.java . . . 901

22 Xml Transform Dom **902**
22.1 DOMLocator.java . . . 902
22.2 DOMResult.java . . . 903
22.3 DOMSource.java . . . 911

23 Xml Transform Sax **913**
23.1 SAXResult.java . . . 913
23.2 SAXSource.java . . . 916
23.3 SAXTransformerFactory.java . . . 920
23.4 TemplatesHandler.java . . . 923
23.5 TransformerHandler.java . . . 924

24 Xml Transform Stax **926**
24.1 StAXResult.java . . . 926
24.2 StAXSource.java . . . 929

25 Xml Transform Stream **934**
25.1 StreamResult.java . . . 934
25.2 StreamSource.java . . . 938

26 Xml Validation **943**
26.1 Schema.java . . . 943
26.2 SchemaFactory.java . . . 945
26.3 SchemaFactoryConfigurationError.java . . . 961
26.4 SchemaFactoryFinder.java . . . 963
26.5 SchemaFactoryLoader.java . . . 971
26.6 SecuritySupport.java . . . 973
26.7 TypeInfoProvider.java . . . 976
26.8 Validator.java . . . 980
26.9 ValidatorHandler.java . . . 990

27 Xml Ws **999**
27.1 Action.java . . . 999
27.2 AsyncHandler.java . . . 1002
27.3 Binding.java . . . 1003
27.4 BindingProvider.java . . . 1004
27.5 BindingType.java . . . 1008
27.6 Dispatch.java . . . 1009
27.7 Endpoint.java . . . 1011
27.8 EndpointContext.java . . . 1021
27.9 EndpointReference.java . . . 1022
27.10 FaultAction.java . . . 1026
27.11 Holder.java . . . 1029
27.12 LogicalMessage.java . . . 1030
27.13 ProtocolException.java . . . 1032
27.14 Provider.java . . . 1034
27.15 RequestWrapper.java . . . 1035
27.16 RespectBinding.java . . . 1036
27.17 RespectBindingFeature.java . . . 1038
27.18 Response.java . . . 1040
27.19 ResponseWrapper.java . . . 1041
27.20 Service.java . . . 1043
27.21 ServiceMode.java . . . 1057
27.22 WebEndpoint.java . . . 1058
27.23 WebFault.java . . . 1059
27.24 WebServiceClient.java . . . 1061
27.25 WebServiceContext.java . . . 1062
27.26 WebServiceException.java . . . 1065
27.27 WebServiceFeature.java . . . 1066
27.28 WebServicePermission.java . . . 1068
27.29 WebServiceProvider.java . . . 1070
27.30 WebServiceRef.java . . . 1071
27.31 WebServiceRefs.java . . . 1074

28 Xml Ws Handler **1075**
28.1 Handler.java . . . 1075
28.2 HandlerResolver.java . . . 1077
28.3 LogicalHandler.java . . . 1078
28.4 LogicalMessageContext.java . . . 1079

28.5 MessageContext.java 1080
28.6 PortInfo.java 1084

29 Xml Ws Handler Soap **1085**
29.1 SOAPHandler.java 1085
29.2 SOAPMessageContext.java 1086

30 Xml Ws Http **1088**
30.1 HTTPBinding.java 1088
30.2 HTTPException.java 1089

31 Xml Ws Soap **1090**
31.1 Addressing.java 1090
31.2 AddressingFeature.java 1092
31.3 MTOM.java 1098
31.4 MTOMFeature.java 1099
31.5 SOAPBinding.java 1102
31.6 SOAPFaultException.java 1104

32 Xml Ws Spi **1106**
32.1 FactoryFinder.java 1106
32.2 Invoker.java 1110
32.3 Provider.java 1112
32.4 ServiceDelegate.java 1122
32.5 WebServiceFeatureAnnotation.java 1134

33 Xml Ws Spi Http **1135**
33.1 HttpContext.java 1135
33.2 HttpExchange.java 1137
33.3 HttpHandler.java 1144
33.4 package-info.java 1145

34 Xml Ws Wsaddressing **1147**
34.1 W3CEndpointReference.java 1147
34.2 W3CEndpointReferenceBuilder.java 1150
34.3 package-info.java 1157

35 Xml Xpath **1157**
35.1 SecuritySupport.java 1157
35.2 XPath.java 1160
35.3 XPathConstants.java 1167
35.4 XPathException.java 1169
35.5 XPathExpression.java 1172
35.6 XPathExpressionException.java 1176
35.7 XPathFactory.java 1178
35.8 XPathFactoryConfigurationException.java 1186
35.9 XPathFactoryFinder.java 1187
35.10 XPathFunction.java 1196
35.11 XPathFunctionException.java 1197
35.12 XPathFunctionResolver.java 1198
35.13 XPathVariableResolver.java 1200

36 Dom **1201**
36.1 Attr.java . . . 1201
36.2 CDATASection.java . . . 1207
36.3 CharacterData.java . . . 1209
36.4 Comment.java . . . 1212
36.5 DOMConfiguration.java . . . 1213
36.6 DOMError.java . . . 1222
36.7 DOMErrorHandler.java . . . 1225
36.8 DOMException.java . . . 1226
36.9 DOMImplementation.java . . . 1229
36.10 DOMImplementationList.java . . . 1233
36.11 DOMImplementationSource.java . . . 1234
36.12 DOMLocator.java . . . 1236
36.13 DOMStringList.java . . . 1238
36.14 Document.java . . . 1239
36.15 DocumentFragment.java . . . 1256
36.16 DocumentType.java . . . 1258
36.17 Element.java . . . 1260
36.18 Entity.java . . . 1269
36.19 EntityReference.java . . . 1271
36.20 NameList.java . . . 1273
36.21 NamedNodeMap.java . . . 1275
36.22 Node.java . . . 1279
36.23 NodeList.java . . . 1297
36.24 Notation.java . . . 1299
36.25 ProcessingInstruction.java . . . 1300
36.26 Text.java . . . 1302
36.27 TypeInfo.java . . . 1305
36.28 UserDataHandler.java . . . 1310

37 Dom Bootstrap **1312**
37.1 DOMImplementationRegistry.java . . . 1312

38 Dom Css **1320**
38.1 CSS2Properties.java . . . 1320
38.2 CSSCharsetRule.java . . . 1354
38.3 CSSFontFaceRule.java . . . 1356
38.4 CSSImportRule.java . . . 1357
38.5 CSSMediaRule.java . . . 1358
38.6 CSSPageRule.java . . . 1361
38.7 CSSPrimitiveValue.java . . . 1362
38.8 CSSRule.java . . . 1368
38.9 CSSRuleList.java . . . 1371
38.10 CSSStyleDeclaration.java . . . 1372
38.11 CSSStyleRule.java . . . 1376
38.12 CSSStyleSheet.java . . . 1377
38.13 CSSUnknownRule.java . . . 1380
38.14 CSSValue.java . . . 1381
38.15 CSSValueList.java . . . 1383
38.16 Counter.java . . . 1384
38.17 DOMImplementationCSS.java . . . 1386
38.18 DocumentCSS.java . . . 1387
38.19 ElementCSSInlineStyle.java . . . 1389

38.20 RGBColor.java 1390
38.21 Rect.java 1391
38.22 ViewCSS.java 1393

39 Dom Events **1394**
39.1 DocumentEvent.java 1394
39.2 Event.java 1396
39.3 EventException.java 1399
39.4 EventListener.java 1401
39.5 EventTarget.java 1402
39.6 MouseEvent.java 1405
39.7 MutationEvent.java 1408
39.8 UIEvent.java 1411

40 Dom Html **1413**
40.1 HTMLAnchorElement.java 1413
40.2 HTMLAppletElement.java 1415
40.3 HTMLAreaElement.java 1418
40.4 HTMLBRElement.java 1420
40.5 HTMLBaseElement.java 1421
40.6 HTMLBaseFontElement.java 1422
40.7 HTMLBodyElement.java 1424
40.8 HTMLButtonElement.java 1426
40.9 HTMLCollection.java 1428
40.10 HTMLDListElement.java 1429
40.11 HTMLDOMImplementation.java 1431
40.12 HTMLDirectoryElement.java 1432
40.13 HTMLDivElement.java 1433
40.14 HTMLDocument.java 1434
40.15 HTMLElement.java 1438
40.16 HTMLFieldSetElement.java 1440
40.17 HTMLFontElement.java 1441
40.18 HTMLFormElement.java 1442
40.19 HTMLFrameElement.java 1444
40.20 HTMLFrameSetElement.java 1447
40.21 HTMLHRElement.java 1448
40.22 HTMLHeadElement.java 1449
40.23 HTMLHeadingElement.java 1451
40.24 HTMLHtmlElement.java 1452
40.25 HTMLIFrameElement.java 1453
40.26 HTMLImageElement.java 1455
40.27 HTMLInputElement.java 1458
40.28 HTMLIsIndexElement.java 1462
40.29 HTMLLIElement.java 1464
40.30 HTMLLabelElement.java 1465
40.31 HTMLLegendElement.java 1466
40.32 HTMLLinkElement.java 1468
40.33 HTMLMapElement.java 1470
40.34 HTMLMenuElement.java 1471
40.35 HTMLMetaElement.java 1473
40.36 HTMLModElement.java 1474
40.37 HTMLOListElement.java 1475
40.38 HTMLObjectElement.java 1477

40.39 HTMLOptGroupElement.java . . . 1480
40.40 HTMLOptionElement.java . . . 1482
40.41 HTMLParagraphElement.java . . . 1484
40.42 HTMLParamElement.java . . . 1485
40.43 HTMLPreElement.java . . . 1487
40.44 HTMLQuoteElement.java . . . 1488
40.45 HTMLScriptElement.java . . . 1489
40.46 HTMLSelectElement.java . . . 1491
40.47 HTMLStyleElement.java . . . 1494
40.48 HTMLTableCaptionElement.java . . . 1495
40.49 HTMLTableCellElement.java . . . 1496
40.50 HTMLTableColElement.java . . . 1499
40.51 HTMLTableElement.java . . . 1501
40.52 HTMLTableRowElement.java . . . 1505
40.53 HTMLTableSectionElement.java . . . 1508
40.54 HTMLTextAreaElement.java . . . 1510
40.55 HTMLTitleElement.java . . . 1513
40.56 HTMLUListElement.java . . . 1514

41 Dom Ls **1516**
41.1 DOMImplementationLS.java . . . 1516
41.2 LSException.java . . . 1519
41.3 LSInput.java . . . 1520
41.4 LSLoadEvent.java . . . 1525
41.5 LSOutput.java . . . 1526
41.6 LSParser.java . . . 1529
41.7 LSParserFilter.java . . . 1539
41.8 LSProgressEvent.java . . . 1543
41.9 LSResourceResolver.java . . . 1545
41.10 LSSerializer.java . . . 1547
41.11 LSSerializerFilter.java . . . 1556

42 Dom Ranges **1558**
42.1 DocumentRange.java . . . 1558
42.2 Range.java . . . 1560
42.3 RangeException.java . . . 1568

43 Dom Stylesheets **1570**
43.1 DocumentStyle.java . . . 1570
43.2 LinkStyle.java . . . 1571
43.3 MediaList.java . . . 1572
43.4 StyleSheet.java . . . 1574
43.5 StyleSheetList.java . . . 1577

44 Dom Traversal **1578**
44.1 DocumentTraversal.java . . . 1578
44.2 NodeFilter.java . . . 1581
44.3 NodeIterator.java . . . 1584
44.4 TreeWalker.java . . . 1587

45 Dom Views **1591**
45.1 AbstractView.java . . . 1591
45.2 DocumentView.java . . . 1592

46 Dom Xpath **1593**
46.1 XPathEvaluator.java 1593
46.2 XPathException.java 1596
46.3 XPathExpression.java 1598
46.4 XPathNSResolver.java 1600
46.5 XPathNamespace.java 1601
46.6 XPathResult.java 1603

47 Sax **1608**
47.1 AttributeList.java 1608
47.2 Attributes.java 1612
47.3 ContentHandler.java 1617
47.4 DTDHandler.java 1626
47.5 DocumentHandler.java 1628
47.6 EntityResolver.java 1633
47.7 ErrorHandler.java 1636
47.8 HandlerBase.java 1639
47.9 InputSource.java 1647
47.10 Locator.java 1653
47.11 Parser.java 1657
47.12 SAXException.java 1661
47.13 SAXNotRecognizedException.java 1665
47.14 SAXNotSupportedException.java 1666
47.15 SAXParseException.java 1668
47.16 XMLFilter.java 1674
47.17 XMLReader.java 1676

48 Sax Ext **1684**
48.1 Attributes2.java 1684
48.2 Attributes2Impl.java 1687
48.3 DeclHandler.java 1693
48.4 DefaultHandler2.java 1696
48.5 EntityResolver2.java 1699
48.6 LexicalHandler.java 1704
48.7 Locator2.java 1708
48.8 Locator2Impl.java 1710

49 Sax Helpers **1713**
49.1 AttributeListImpl.java 1713
49.2 AttributesImpl.java 1719
49.3 DefaultHandler.java 1731
49.4 LocatorImpl.java 1741
49.5 NamespaceSupport.java 1745
49.6 NewInstance.java 1761
49.7 ParserAdapter.java 1763
49.8 ParserFactory.java 1783
49.9 SecuritySupport.java 1786
49.10 XMLFilterImpl.java 1788
49.11 XMLReaderAdapter.java 1802
49.12 XMLReaderFactory.java 1813

1 Xml (javax.xml package)

1.1 XMLConstants.java

Secuencia 2: javax/xml/XMLConstants.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml;

/**
 * <p>Utility class to contain basic XML values as constants.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @version $Revision: 1.8 $, $Date: 2010/05/25 16:19:45 $
 * @see <a href="http://www.w3.org/TR/xml11/">Extensible Markup Language (XML) 1.1</a>
 * @see <a href="http://www.w3.org/TR/REC-xml">Extensible Markup Language (XML) 1.0 (Second Edition
     )</a>
 * @see <a href="http://www.w3.org/XML/xml-V10-2e-errata">XML 1.0 Second Edition Specification
     Errata</a>
 * @see <a href="http://www.w3.org/TR/xml-names11/">Namespaces in XML 1.1</a>
 * @see <a href="http://www.w3.org/TR/REC-xml-names">Namespaces in XML</a>
 * @see <a href="http://www.w3.org/XML/xml-names-19990114-errata">Namespaces in XML Errata</a>
 * @see <a href="http://www.w3.org/TR/xmlschema-1/">XML Schema Part 1: Structures</a>
 * @since 1.5
 **/

public final class XMLConstants {

    /**
```

```
     * <p>Private constructor to prevent instantiation.</p>
     */
        private XMLConstants() {
        }

    /**
     * <p>Namespace URI to use to represent that there is no Namespace.</p>
     *
     * <p>Defined by the Namespace specification to be "".</p>
     *
     * @see <a href="http://www.w3.org/TR/REC-xml-names/#defaulting">
     * Namespaces in XML, 5.2 Namespace Defaulting</a>
     */
    public static final String NULL_NS_URI = "";

    /**
     * <p>Prefix to use to represent the default XML Namespace.</p>
     *
     * <p>Defined by the XML specification to be "".</p>
     *
     * @see <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
     * Namespaces in XML, 3. Qualified Names</a>
     */
    public static final String DEFAULT_NS_PREFIX = "";

    /**
     * <p>The official XML Namespace name URI.</p>
     *
     * <p>Defined by the XML specification to be
     * "{@code http://www.w3.org/XML/1998/namespace}".</p>
     *
     * @see <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
     * Namespaces in XML, 3. Qualified Names</a>
     */
    public static final String XML_NS_URI =
        "http://www.w3.org/XML/1998/namespace";

    /**
     * <p>The official XML Namespace prefix.</p>
     *
     * <p>Defined by the XML specification to be "{@code xml}".</p>
     *
     * @see <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
     * Namespaces in XML, 3. Qualified Names<</a>
     */
    public static final String XML_NS_PREFIX = "xml";

    /**
     * <p>The official XML attribute used for specifying XML Namespace
     * declarations, {@link #XMLNS_ATTRIBUTE
```

```
     * XMLConstants.XMLNS_ATTRIBUTE}, Namespace name URI.</p>
     *
     * <p>Defined by the XML specification to be
     * "{@code http://www.w3.org/2000/xmlns/}".</p>
     *
     * @see <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
     * Namespaces in XML, 3. Qualified Names</a>
     * @see <a
     * href="http://www.w3.org/XML/xml-names-19990114-errata">
     * Namespaces in XML Errata</a>
     */
    public static final String XMLNS_ATTRIBUTE_NS_URI =
        "http://www.w3.org/2000/xmlns/";

    /**
     * <p>The official XML attribute used for specifying XML Namespace
     * declarations.</p>
     *
     * <p>It is <strong><em>NOT</em></strong> valid to use as a
     * prefix.  Defined by the XML specification to be
     * "{@code xmlns}".</p>
     *
     * @see <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
     * Namespaces in XML, 3. Qualified Names</a>
     */
    public static final String XMLNS_ATTRIBUTE = "xmlns";

    /**
     * <p>W3C XML Schema Namespace URI.</p>
     *
     * <p>Defined to be "{@code http://www.w3.org/2001/XMLSchema}".
     *
     * @see <a href=
     *  "http://www.w3.org/TR/xmlschema-1/#Instance_Document_Constructions">
     *  XML Schema Part 1:
     *  Structures, 2.6 Schema-Related Markup in Documents Being Validated</a>
     */
    public static final String W3C_XML_SCHEMA_NS_URI =
        "http://www.w3.org/2001/XMLSchema";

    /**
     * <p>W3C XML Schema Instance Namespace URI.</p>
     *
     * <p>Defined to be "{@code http://www.w3.org/2001/XMLSchema-instance}".</p>
     *
     * @see <a href=
     *  "http://www.w3.org/TR/xmlschema-1/#Instance_Document_Constructions">
     *  XML Schema Part 1:
     *  Structures, 2.6 Schema-Related Markup in Documents Being Validated</a>
     */
    public static final String W3C_XML_SCHEMA_INSTANCE_NS_URI =
```

```
        "http://www.w3.org/2001/XMLSchema-instance";

        /**
         * <p>W3C XPath Datatype Namespace URI.</p>
         *
         * <p>Defined to be "{@code http://www.w3.org/2003/11/xpath-datatypes}".</p>
         *
         * @see <a href="http://www.w3.org/TR/xpath-datamodel">XQuery 1.0 and XPath 2.0 Data Model</a>
         */
        public static final String W3C_XPATH_DATATYPE_NS_URI = "http://www.w3.org/2003/11/xpath-datatypes";

    /**
     * <p>XML Document Type Declaration Namespace URI as an arbitrary value.</p>
     *
     * <p>Since not formally defined by any existing standard, arbitrarily define to be "{@code http://www.w3.org/TR/REC-xml}".
     */
    public static final String XML_DTD_NS_URI = "http://www.w3.org/TR/REC-xml";

        /**
         * <p>RELAX NG Namespace URI.</p>
         *
         * <p>Defined to be "{@code http://relaxng.org/ns/structure/1.0}".</p>
         *
         * @see <a href="http://relaxng.org/spec-20011203.html">RELAX NG Specification</a>
         */
        public static final String RELAXNG_NS_URI = "http://relaxng.org/ns/structure/1.0";

        /**
         * <p>Feature for secure processing.</p>
         *
         * <ul>
         *   <li>
         *     {@code true} instructs the implementation to process XML securely.
         *     This may set limits on XML constructs to avoid conditions such as denial of service attacks.
         *   </li>
         *   <li>
         *     {@code false} instructs the implementation to process XML in accordance with the XML specifications
         *     ignoring security issues such as limits on XML constructs to avoid conditions such as denial of service attacks.
         *   </li>
         * </ul>
         */
        public static final String FEATURE_SECURE_PROCESSING = "http://javax.xml.XMLConstants/feature/secure-processing";


        /**
         * <p>Property: accessExternalDTD</p>
```

```
        *
        * <p>
        * Restrict access to external DTDs and external Entity References to the protocols specified.
        * If access is denied due to the restriction of this property, a runtime exception that
        * is specific to the context is thrown. In the case of {@link javax.xml.parsers.SAXParser}
        * for example, {@link org.xml.sax.SAXException} is thrown.
        * </p>
        *
        * <p>
        * <b>Value: </b> a list of protocols separated by comma. A protocol is the scheme portion of a
        * {@link java.net.URI}, or in the case of the JAR protocol, "jar" plus the scheme portion
        * separated by colon.
        * A scheme is defined as:
        *
        * <blockquote>
        * scheme = alpha *( alpha | digit | "+" | "-" | "." )<br>
        * where alpha = a-z and A-Z.<br><br>
        *
        * And the JAR protocol:<br>
        *
        * jar[:scheme]<br><br>
        *
        * Protocols including the keyword "jar" are case-insensitive. Any whitespaces as defined by
        * {@link java.lang.Character#isSpaceChar } in the value will be ignored.
        * Examples of protocols are file, http, jar:file.
        *
        * </blockquote>
        *</p>
        *
        *<p>
        * <b>Default value:</b> The default value is implementation specific and therefore not specified.
        * The following options are provided for consideration:
        * <blockquote>
        * <UL>
        *     <LI>an empty string to deny all access to external references;</LI>
        *     <LI>a specific protocol, such as file, to give permission to only the protocol;</LI>
        *     <LI>the keyword "all" to grant  permission to all protocols.</LI>
        *</UL><br>
        *      When FEATURE_SECURE_PROCESSING is enabled,  it is recommended that implementations
        *      restrict external connections by default, though this may cause problems for applications
        *      that process XML/XSD/XSL with external references.
        * </blockquote>
        * </p>
        *
        * <p>
        * <b>Granting all access:</b>  the keyword "all" grants permission to all protocols.
        * </p>
        * <p>
```

```
     * <b>System Property:</b> The value of this property can be set or overridden by
     * system property {@code javax.xml.accessExternalDTD}.
     * </p>
     *
     * <p>
     * <b>${JAVA_HOME}/lib/jaxp.properties:</b> This configuration file is in standard
     * {@link java.util.Properties} format. If the file exists and the system property is specified,
     * its value will be used to override the default of the property.
     * </p>
     *
     * <p>
     *
     * </p>
     * @since 1.7
     */
    public static final String ACCESS_EXTERNAL_DTD = "http://javax.xml.XMLConstants/property/accessExternalDTD";

    /**
     * <p>Property: accessExternalSchema</p>
     *
     * <p>
     * Restrict access to the protocols specified for external reference set by the
     * schemaLocation attribute, Import and Include element. If access is denied
     * due to the restriction of this property, a runtime exception that is specific
     * to the context is thrown. In the case of {@link javax.xml.validation.SchemaFactory}
     * for example, org.xml.sax.SAXException is thrown.
     * </p>
     * <p>
     * <b>Value:</b> a list of protocols separated by comma. A protocol is the scheme portion of a
     * {@link java.net.URI}, or in the case of the JAR protocol, "jar" plus the scheme portion
     * separated by colon.
     * A scheme is defined as:
     *
     * <blockquote>
     * scheme = alpha *( alpha | digit | "+" | "-" | "." )<br>
     * where alpha = a-z and A-Z.<br><br>
     *
     * And the JAR protocol:<br>
     *
     * jar[:scheme]<br><br>
     *
     * Protocols including the keyword "jar" are case-insensitive. Any whitespaces as defined by
     * {@link java.lang.Character#isSpaceChar } in the value will be ignored.
     * Examples of protocols are file, http, jar:file.
     *
     * </blockquote>
     *</p>
     *
     *<p>
```

```
295         * <b>Default value:</b> The default value is implementation specific and therefore not
                specified.
296         * The following options are provided for consideration:
297         * <blockquote>
298         * <UL>
299         *     <LI>an empty string to deny all access to external references;</LI>
300         *     <LI>a specific protocol, such as file, to give permission to only the protocol;</LI>
301         *     <LI>the keyword "all" to grant  permission to all protocols.</LI>
302         *</UL><br>
303         *      When FEATURE_SECURE_PROCESSING is enabled,  it is recommended that implementations
304         *      restrict external connections by default, though this may cause problems for
                applications
305         *      that process XML/XSD/XSL with external references.
306         * </blockquote>
307         * </p>
308         * <p>
309         * <b>Granting all access:</b>  the keyword "all" grants permission to all protocols.
310         * </p>
311         *
312         * <p>
313         * <b>System Property:</b> The value of this property can be set or overridden by
314         * system property {@code javax.xml.accessExternalSchema}
315         * </p>
316         *
317         * <p>
318         * <b>${JAVA_HOME}/lib/jaxp.properties:</b> This configuration file is in standard
319         * java.util.Properties format. If the file exists and the system property is specified,
320         * its value will be used to override the default of the property.
321         *
322         * @since 1.7
323         * </p>
324         */
325        public static final String ACCESS_EXTERNAL_SCHEMA = "http://javax.xml.XMLConstants/property
               /accessExternalSchema";
326
327        /**
328         * <p>Property: accessExternalStylesheet</p>
329         *
330         * <p>
331         * Restrict access to the protocols specified for external references set by the
332         * stylesheet processing instruction, Import and Include element, and document function.
333         * If access is denied due to the restriction of this property, a runtime exception
334         * that is specific to the context is thrown. In the case of constructing new
335         * {@link javax.xml.transform.Transformer} for example,
336         * {@link javax.xml.transform.TransformerConfigurationException}
337         * will be thrown by the {@link javax.xml.transform.TransformerFactory}.
338         * </p>
339         * <p>
340         * <b>Value:</b> a list of protocols separated by comma. A protocol is the scheme portion
                of a
341         * {@link java.net.URI}, or in the case of the JAR protocol, "jar" plus the scheme portion
342         * separated by colon.
343         * A scheme is defined as:
```

```
         *
         * <blockquote>
         * scheme = alpha *( alpha | digit | "+" | "-" | "." )<br>
         * where alpha = a-z and A-Z.<br><br>
         *
         * And the JAR protocol:<br>
         *
         * jar[:scheme]<br><br>
         *
         * Protocols including the keyword "jar" are case-insensitive. Any whitespaces as defined by
         * {@link java.lang.Character#isSpaceChar } in the value will be ignored.
         * Examples of protocols are file, http, jar:file.
         *
         * </blockquote>
         *</p>
         *
         *<p>
         * <b>Default value:</b> The default value is implementation specific and therefore not specified.
         * The following options are provided for consideration:
         * <blockquote>
         * <UL>
         *     <LI>an empty string to deny all access to external references;</LI>
         *     <LI>a specific protocol, such as file, to give permission to only the protocol;</LI>
         *     <LI>the keyword "all" to grant  permission to all protocols.</LI>
         *</UL><br>
         *      When FEATURE_SECURE_PROCESSING is enabled,  it is recommended that implementations
         *      restrict external connections by default, though this may cause problems for applications
         *      that process XML/XSD/XSL with external references.
         * </blockquote>
         * </p>
         * <p>
         * <b>Granting all access:</b>  the keyword "all" grants permission to all protocols.
         * </p>
         *
         * <p>
         * <b>System Property:</b> The value of this property can be set or overridden by
         * system property {@code javax.xml.accessExternalStylesheet}
         * </p>
         *
         * <p>
         * <b>${JAVA_HOME}/lib/jaxp.properties: </b> This configuration file is in standard
         * java.util.Properties format. If the file exists and the system property is specified,
         * its value will be used to override the default of the property.
         *
         * @since 1.7
         */
        public static final String ACCESS_EXTERNAL_STYLESHEET = "http://javax.xml.XMLConstants/property/accessExternalStylesheet";

}
```

2 Xml Bind (javax.xml.bind package)

2.1 Binder.java

Secuencia 3: javax/xml/bind/Binder.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import org.w3c.dom.Node;

import javax.xml.validation.Schema;

/**
 * Enable synchronization between XML infoset nodes and JAXB objects
 * representing same XML document.
 *
 * <p>
 * An instance of this class maintains the association between XML nodes of
 * an infoset preserving view and a JAXB representation of an XML document.
 * Navigation between the two views is provided by the methods
 * {@link #getXMLNode(Object)} and {@link #getJAXBNode(Object)}.
 *
 * <p>
 * Modifications can be made to either the infoset preserving view or the
 * JAXB representation of the document while the other view remains
 * unmodified. The binder is able to synchronize the changes made in the
 * modified view back into the other view using the appropriate
 * Binder update methods, {@link #updateXML(Object, Object)} or
```

```
 * {@link #updateJAXB(Object)}.
 *
 * <p>
 * A typical usage scenario is the following:
 * <ul>
 *   <li>load XML document into an XML infoset representation</li>
 *   <li>{@link #unmarshal(Object)} XML infoset view to JAXB view.
 *       (Note to conserve resources, it is possible to only unmarshal a
 *       subtree of the XML infoset view to the JAXB view.)</li>
 *   <li>application access/updates JAXB view of XML document.</li>
 *   <li>{@link #updateXML(Object)} synchronizes modifications to JAXB view
 *       back into the XML infoset view. Update operation preserves as
 *       much of original XML infoset as possible (i.e. comments, PI, ...)</li>
 * </ul>
 *
 * <p>
 * A Binder instance is created using the factory method
 * {@link JAXBContext#createBinder()} or {@link JAXBContext#createBinder(Class)}.
 *
 * <p>
 * The template parameter, <code>XmlNode</code>, is the
 * root interface/class for the XML infoset preserving representation.
 * A Binder implementation is required to minimally support
 * an <code>XmlNode</code> value of <code>org.w3c.dom.Node.class</code>.
 * A Binder implementation can support alternative XML infoset
 * preserving representations.
 *
 * @author
 *     Kohsuke Kawaguchi (kohsuke.kawaguchi@sun.com)
 *     Joseph Fialli
 *
 * @since JAXB 2.0
 */
public abstract class Binder<XmlNode> {
    /**
     * Unmarshal XML infoset view to a JAXB object tree.
     *
     * <p>
     * This method is similar to {@link Unmarshaller#unmarshal(Node)}
     * with the addition of maintaining the association between XML nodes
     * and the produced JAXB objects, enabling future update operations,
     * {@link #updateXML(Object, Object)} or {@link #updateJAXB(Object)}.
     *
     * <p>
     * When {@link #getSchema()} is non-null, <code>xmlNode</code>
     * and its descendants is validated during this operation.
     *
     * <p>
     * This method throws {@link UnmarshalException} when the Binder's
     * {@link JAXBContext} does not have a mapping for the XML element name
     * or the type, specifiable via <tt>@xsi:type</tt>, of <tt>xmlNode</tt>
     * to a JAXB mapped class. The method {@link #unmarshal(Object, Class)}
     * enables an application to specify the JAXB mapped class that
```

```
     * the <tt>xmlNode</tt> should be mapped to.
     *
     * @param xmlNode
     *      the document/element to unmarshal XML data from.
     *
     * @return
     *      the newly created root object of the JAXB object tree.
     *
     * @throws JAXBException
     *      If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Binder</tt> is unable to perform the XML to Java
     *     binding.
     * @throws IllegalArgumentException
     *      If the node parameter is null
     */
    public abstract Object unmarshal( XmlNode xmlNode ) throws JAXBException;

    /**
     * Unmarshal XML root element by provided <tt>declaredType</tt>
     * to a JAXB object tree.
     *
     * <p>
     * Implements <a href="Unmarshaller.html#unmarshalByDeclaredType">Unmarshal by Declared Type</a>
     *
     * <p>
     * This method is similar to {@link Unmarshaller#unmarshal(Node, Class)}
     * with the addition of maintaining the association between XML nodes
     * and the produced JAXB objects, enabling future update operations,
     * {@link #updateXML(Object, Object)} or {@link #updateJAXB(Object)}.
     *
     * <p>
     * When {@link #getSchema()} is non-null, <code>xmlNode</code>
     * and its descendants is validated during this operation.
     *
     * @param xmlNode
     *      the document/element to unmarshal XML data from.
     * @param declaredType
     *      appropriate JAXB mapped class to hold <tt>node</tt>'s XML data.
     *
     * @return
     * <a href="JAXBElement.html">JAXB Element</a> representation
     * of <tt>node</tt>
     *
     * @throws JAXBException
     *      If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Binder</tt> is unable to perform the XML to Java
```

```
     *     binding.
     * @throws IllegalArgumentException
     *      If any of the input parameters are null
     * @since JAXB2.0
     */
    public abstract <T> JAXBElement<T>
        unmarshal( XmlNode xmlNode, Class<T> declaredType )
        throws JAXBException;

    /**
     * Marshal a JAXB object tree to a new XML document.
     *
     * <p>
     * This method is similar to {@link Marshaller#marshal(Object, Node)}
     * with the addition of maintaining the association between JAXB objects
     * and the produced XML nodes,
     * enabling future update operations such as
     * {@link #updateXML(Object, Object)} or {@link #updateJAXB(Object)}.
     *
     * <p>
     * When {@link #getSchema()} is non-null, the marshalled
     * xml content is validated during this operation.
     *
     * @param jaxbObject
     *      The content tree to be marshalled.
     * @param xmlNode
     *      The parameter must be a Node that accepts children.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Binder</tt> is unable to marshal <tt>jaxbObject</tt> (or any
     *      object reachable from <tt>jaxbObject</tt>).
     *
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public abstract void marshal( Object jaxbObject, XmlNode xmlNode ) throws JAXBException;

    /**
     * Gets the XML element associated with the given JAXB object.
     *
     * <p>
     * Once a JAXB object tree is associated with an XML fragment,
     * this method enables navigation between the two trees.
     *
     * <p>
     * An association between an XML element and a JAXB object is
     * established by the bind methods and the update methods.
     * Note that this association is partial; not all XML elements
     * have associated JAXB objects, and not all JAXB objects have
```

```
     * associated XML elements.
     *
     * @param jaxbObject An instance that is reachable from a prior
     *                   call to a bind or update method that returned
     *                   a JAXB object tree.
     *
     * @return
     *      null if the specified JAXB object is not known to this
     *      {@link Binder}, or if it is not associated with an
     *      XML element.
     *
     * @throws IllegalArgumentException
     *      If the jaxbObject parameter is null
     */
    public abstract XmlNode getXMLNode( Object jaxbObject );

    /**
     * Gets the JAXB object associated with the given XML element.
     *
     * <p>
     * Once a JAXB object tree is associated with an XML fragment,
     * this method enables navigation between the two trees.
     *
     * <p>
     * An association between an XML element and a JAXB object is
     * established by the unmarshal, marshal and update methods.
     * Note that this association is partial; not all XML elements
     * have associated JAXB objects, and not all JAXB objects have
     * associated XML elements.
     *
     * @return
     *      null if the specified XML node is not known to this
     *      {@link Binder}, or if it is not associated with a
     *      JAXB object.
     *
     * @throws IllegalArgumentException
     *      If the node parameter is null
     */
    public abstract Object getJAXBNode( XmlNode xmlNode );

    /**
     * Takes an JAXB object and updates
     * its associated XML node and its descendants.
     *
     * <p>
     * This is a convenience method of:
     * <pre>
     * updateXML( jaxbObject, getXMLNode(jaxbObject));
     * </pre>
     *
     * @throws JAXBException
     *      If any unexpected problem occurs updating corresponding XML content.
     * @throws IllegalArgumentException
```

```
     *      If the jaxbObject parameter is null
     */
    public abstract XmlNode updateXML( Object jaxbObject ) throws JAXBException;

    /**
     * Changes in JAXB object tree are updated in its associated XML parse tree.
     *
     * <p>
     * This operation can be thought of as an "in-place" marshalling.
     * The difference is that instead of creating a whole new XML tree,
     * this operation updates an existing tree while trying to preserve
     * the XML as much as possible.
     *
     * <p>
     * For example, unknown elements/attributes in XML that were not bound
     * to JAXB will be left untouched (whereas a marshalling operation
     * would create a new tree that doesn't contain any of those.)
     *
     * <p>
     * As a side-effect, this operation updates the association between
     * XML nodes and JAXB objects.
     *
     * @param jaxbObject root of potentially modified JAXB object tree
     * @param xmlNode    root of update target XML parse tree
     *
     * @return
     *      Returns the updated XML node. Typically, this is the same
     *      node you passed in as <i>xmlNode</i>, but it maybe
     *      a different object, for example when the tag name of the object
     *      has changed.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs updating corresponding XML content.
     * @throws IllegalArgumentException
     *      If any of the input parameters are null
     */
    public abstract XmlNode updateXML( Object jaxbObject, XmlNode xmlNode ) throws JAXBException;

    /**
     * Takes an XML node and updates its associated JAXB object and its descendants.
     *
     * <p>
     * This operation can be thought of as an "in-place" unmarshalling.
     * The difference is that instead of creating a whole new JAXB tree,
     * this operation updates an existing tree, reusing as much JAXB objects
     * as possible.
     *
     * <p>
     * As a side-effect, this operation updates the association between
     * XML nodes and JAXB objects.
     *
     * @return
     *      Returns the updated JAXB object. Typically, this is the same
```

```
     *      object that was returned from earlier
     *      {@link #marshal(Object,Object)} or
     *      {@link #updateJAXB(Object)} method invocation,
     *      but it maybe
     *      a different object, for example when the name of the XML
     *      element has changed.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs updating corresponding JAXB mapped content.
     * @throws IllegalArgumentException
     *      If node parameter is null
     */
    public abstract Object updateJAXB( XmlNode xmlNode ) throws JAXBException;


    /**
     * Specifies whether marshal, unmarshal and update methods
     * performs validation on their XML content.
     *
     * @param schema set to null to disable validation.
     *
     * @see Unmarshaller#setSchema(Schema)
     */
    public abstract void setSchema( Schema schema );

    /**
     * Gets the last {@link Schema} object (including null) set by the
     * {@link #setSchema(Schema)} method.
     *
     * @return the Schema object for validation or null if not present
     */
    public abstract Schema getSchema();

    /**
     * Allow an application to register a <tt>ValidationEventHandler</tt>.
     * <p>
     * The <tt>ValidationEventHandler</tt> will be called by the JAXB Provider
     * if any validation errors are encountered during calls to any of the
     * Binder unmarshal, marshal and update methods.
     *
     * <p>
     * Calling this method with a null parameter will cause the Binder
     * to revert back to the default default event handler.
     *
     * @param handler the validation event handler
     * @throws JAXBException if an error was encountered while setting the
     *         event handler
     */
    public abstract void setEventHandler( ValidationEventHandler handler ) throws JAXBException;

    /**
     * Return the current event handler or the default event handler if one
     * hasn't been set.
```

```
     *
     * @return the current ValidationEventHandler or the default event handler
     *         if it hasn't been set
     * @throws JAXBException if an error was encountered while getting the
     *         current event handler
     */
    public abstract ValidationEventHandler getEventHandler() throws JAXBException;

    /**
     *
     * Set the particular property in the underlying implementation of
     * <tt>Binder</tt>.  This method can only be used to set one of
     * the standard JAXB defined unmarshal/marshal properties
     * or a provider specific property for binder, unmarshal or marshal.
     * Attempting to set an undefined property will result in
     * a PropertyException being thrown.  See
     * <a href="Unmarshaller.html#supportedProps">Supported Unmarshal Properties</a>
     * and
     * <a href="Marshaller.html#supportedProps">Supported Marshal Properties</a>.
     *
     * @param name the name of the property to be set. This value can either
     *              be specified using one of the constant fields or a user
     *              supplied string.
     * @param value the value of the property to be set
     *
     * @throws PropertyException when there is an error processing the given
     *                           property or value
     * @throws IllegalArgumentException
     *      If the name parameter is null
     */
    abstract public void setProperty( String name, Object value ) throws PropertyException;


    /**
     * Get the particular property in the underlying implementation of
     * <tt>Binder</tt>.  This method can only
     * be used to get one of
     * the standard JAXB defined unmarshal/marshal properties
     * or a provider specific property for binder, unmarshal or marshal.
     * Attempting to get an undefined property will result in
     * a PropertyException being thrown.  See
     * <a href="Unmarshaller.html#supportedProps">Supported Unmarshal Properties</a>
     * and
     * <a href="Marshaller.html#supportedProps">Supported Marshal Properties</a>.
     *
     * @param name the name of the property to retrieve
     * @return the value of the requested property
     *
     * @throws PropertyException
     *      when there is an error retrieving the given property or value
     *      property name
     * @throws IllegalArgumentException
     *      If the name parameter is null
```

```
     */
    abstract public Object getProperty( String name ) throws PropertyException;

}
```

2.2 ContextFinder.java

Secuencia 4: javax/xml/bind/ContextFinder.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import java.util.Iterator;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.net.URL;
import java.util.Map;
import java.util.Properties;
import java.util.StringTokenizer;
import java.util.logging.ConsoleHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.security.AccessController;

import static javax.xml.bind.JAXBContext.JAXB_CONTEXT_FACTORY;
```

```java


/**
 * This class is package private and therefore is not exposed as part of the
 * JAXB API.
 *
 * This code is designed to implement the JAXB 1.0 spec pluggability feature
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 */
class ContextFinder {
    private static final Logger logger;
    static {
        logger = Logger.getLogger("javax.xml.bind");
        try {
            if (AccessController.doPrivileged(new GetPropertyAction("jaxb.debug")) != null) {
                // disconnect the logger from a bigger framework (if any)
                // and take the matters into our own hands
                logger.setUseParentHandlers(false);
                logger.setLevel(Level.ALL);
                ConsoleHandler handler = new ConsoleHandler();
                handler.setLevel(Level.ALL);
                logger.addHandler(handler);
            } else {
                // don't change the setting of this logger
                // to honor what other frameworks
                // have done on configurations.
            }
        } catch(Throwable t) {
            // just to be extra safe. in particular System.getProperty may throw
            // SecurityException.
        }
    }

    /**
     * If the {@link InvocationTargetException} wraps an exception that shouldn't be wrapped,
     * throw the wrapped exception.
     */
    private static void handleInvocationTargetException(InvocationTargetException x) throws
        JAXBException {
        Throwable t = x.getTargetException();
        if( t != null ) {
            if( t instanceof JAXBException )
                // one of our exceptions, just re-throw
                throw (JAXBException)t;
            if( t instanceof RuntimeException )
                // avoid wrapping exceptions unnecessarily
                throw (RuntimeException)t;
            if( t instanceof Error )
                throw (Error)t;
        }
    }
```

```


    /**
     * Determine if two types (JAXBContext in this case) will generate a ClassCastException.
     *
     * For example, (targetType)originalType
     *
     * @param originalType
     *          The Class object of the type being cast
     * @param targetType
     *          The Class object of the type that is being cast to
     * @return JAXBException to be thrown.
     */
    private static JAXBException handleClassCastException(Class originalType, Class targetType) {
        final URL targetTypeURL = which(targetType);

        return new JAXBException(Messages.format(Messages.ILLEGAL_CAST,
                // we don't care where the impl class is, we want to know where JAXBContext lives
                    in the impl
                // class' ClassLoader
                getClassClassLoader(originalType).getResource("javax/xml/bind/JAXBContext.class"),
                targetTypeURL));
    }

    /**
     * Create an instance of a class using the specified ClassLoader
     */
    static JAXBContext newInstance( String contextPath,
                                    String className,
                                    ClassLoader classLoader,
                                    Map properties )
        throws JAXBException {
        try {
            Class spFactory = safeLoadClass(className,classLoader);
            return newInstance(contextPath, spFactory, classLoader, properties);
        } catch (ClassNotFoundException x) {
            throw new JAXBException(
                Messages.format( Messages.PROVIDER_NOT_FOUND, className ),
                x);
        } catch (RuntimeException x) {
            // avoid wrapping RuntimeException to JAXBException,
            // because it indicates a bug in this code.
            throw x;
        } catch (Exception x) {
            // can't catch JAXBException because the method is hidden behind
            // reflection.  Root element collisions detected in the call to
            // createContext() are reported as JAXBExceptions - just re-throw it
            // some other type of exception - just wrap it
            throw new JAXBException(
                Messages.format( Messages.COULD_NOT_INSTANTIATE, className, x ),
                x);
        }
    }
```

```java

    static JAXBContext newInstance( String contextPath,
                                    Class spFactory,
                                    ClassLoader classLoader,
                                    Map properties )
        throws JAXBException
    {
        try {
            /*
             * javax.xml.bind.context.factory points to a class which has a
             * static method called 'createContext' that
             * returns a javax.xml.JAXBContext.
             */

            Object context = null;

            // first check the method that takes Map as the third parameter.
            // this is added in 2.0.
            try {
                Method m = spFactory.getMethod("createContext",String.class,ClassLoader.class,Map.class);
                // any failure in invoking this method would be considered fatal
                context = m.invoke(null,contextPath,classLoader,properties);
            } catch (NoSuchMethodException e) {
                // it's not an error for the provider not to have this method.
            }

            if(context==null) {
                // try the old method that doesn't take properties. compatible with 1.0.
                // it is an error for an implementation not to have both forms of the createContext method.
                Method m = spFactory.getMethod("createContext",String.class,ClassLoader.class);
                // any failure in invoking this method would be considered fatal
                context = m.invoke(null,contextPath,classLoader);
            }

            if(!(context instanceof JAXBContext)) {
                // the cast would fail, so generate an exception with a nice message
                throw handleClassCastException(context.getClass(), JAXBContext.class);
            }
            return (JAXBContext)context;
        } catch (InvocationTargetException x) {
            handleInvocationTargetException(x);
            // for other exceptions, wrap the internal target exception
            // with a JAXBException
            Throwable e = x;
            if(x.getTargetException()!=null)
                e = x.getTargetException();

            throw new JAXBException( Messages.format( Messages.COULD_NOT_INSTANTIATE, spFactory, e ), e );
        } catch (RuntimeException x) {
            // avoid wrapping RuntimeException to JAXBException,
```

```
            // because it indicates a bug in this code.
            throw x;
        } catch (Exception x) {
            // can't catch JAXBException because the method is hidden behind
            // reflection.  Root element collisions detected in the call to
            // createContext() are reported as JAXBExceptions - just re-throw it
            // some other type of exception - just wrap it
            throw new JAXBException(
                Messages.format( Messages.COULD_NOT_INSTANTIATE, spFactory, x ),
                x);
        }
    }


    /**
     * Create an instance of a class using the thread context ClassLoader
     */
    static JAXBContext newInstance(
                            Class[] classes,
                            Map properties,
                            String className) throws JAXBException {
        ClassLoader cl = getContextClassLoader();
        Class spi;
        try {
            spi = safeLoadClass(className,cl);
        } catch (ClassNotFoundException e) {
            throw new JAXBException(e);
        }

        if(logger.isLoggable(Level.FINE)) {
            // extra check to avoid costly which operation if not logged
            logger.log(Level.FINE, "loaded {0} from {1}", new Object[]{className, which(spi)});
        }

        return newInstance(classes, properties, spi);
    }

    static JAXBContext newInstance(Class[] classes,
                                   Map properties,
                                   Class spFactory) throws JAXBException {
        Method m;
        try {
            m = spFactory.getMethod("createContext", Class[].class, Map.class);
        } catch (NoSuchMethodException e) {
            throw new JAXBException(e);
        }
        try {
            Object context = m.invoke(null, classes, properties);
            if(!(context instanceof JAXBContext)) {
                // the cast would fail, so generate an exception with a nice message
                throw handleClassCastException(context.getClass(), JAXBContext.class);
            }
            return (JAXBContext)context;
```

```
        } catch (IllegalAccessException e) {
            throw new JAXBException(e);
        } catch (InvocationTargetException e) {
            handleInvocationTargetException(e);

            Throwable x = e;
            if (e.getTargetException() != null)
                x = e.getTargetException();

            throw new JAXBException(x);
        }
    }

    static JAXBContext find(String factoryId, String contextPath, ClassLoader classLoader, Map
        properties ) throws JAXBException {

        // TODO: do we want/need another layer of searching in $java.home/lib/jaxb.properties like
            JAXP?

        final String jaxbContextFQCN = JAXBContext.class.getName();

        // search context path for jaxb.properties first
        StringBuilder propFileName;
        StringTokenizer packages = new StringTokenizer( contextPath, ":" );
        String factoryClassName;

        if(!packages.hasMoreTokens())
            // no context is specified
            throw new JAXBException(Messages.format(Messages.NO_PACKAGE_IN_CONTEXTPATH));


        logger.fine("Searching jaxb.properties");

        while( packages.hasMoreTokens() ) {
            String packageName = packages.nextToken(":").replace('.','/');
            // com.acme.foo - > com/acme/foo/jaxb.properties
             propFileName = new StringBuilder().append(packageName).append("/jaxb.properties");

            Properties props = loadJAXBProperties( classLoader, propFileName.toString() );
            if (props != null) {
                if (props.containsKey(factoryId)) {
                    factoryClassName = props.getProperty(factoryId);
                    return newInstance( contextPath, factoryClassName, classLoader, properties );
                } else {
                    throw new JAXBException(Messages.format(Messages.MISSING_PROPERTY, packageName,
                         factoryId));
                }
            }
        }

        logger.fine("Searching the system property");

        // search for a system property second (javax.xml.bind.JAXBContext)
```

```java
        factoryClassName = AccessController.doPrivileged(new GetPropertyAction(JAXBContext.JAXB_CONTEXT_FACTORY));
        if(  factoryClassName != null ) {
            return newInstance( contextPath, factoryClassName, classLoader, properties );
        } else { // leave this here to assure compatibility
            factoryClassName = AccessController.doPrivileged(new GetPropertyAction(jaxbContextFQCN));
            if(  factoryClassName != null ) {
                return newInstance( contextPath, factoryClassName, classLoader, properties );
            }
        }

        // OSGi search
        Class jaxbContext = lookupJaxbContextUsingOsgiServiceLoader();
        if (jaxbContext != null) {
            logger.fine("OSGi environment detected");
            return newInstance(contextPath, jaxbContext, classLoader, properties);
        }

        logger.fine("Searching META-INF/services");
        // search META-INF services next
        BufferedReader r = null;
        try {
            final StringBuilder resource = new StringBuilder().append("META-INF/services/").append(jaxbContextFQCN);
            final InputStream resourceStream =
                    classLoader.getResourceAsStream(resource.toString());

            if (resourceStream != null) {
                r = new BufferedReader(new InputStreamReader(resourceStream, "UTF-8"));
                factoryClassName = r.readLine();
                if (factoryClassName != null) {
                    factoryClassName = factoryClassName.trim();
                }
                r.close();
                return newInstance(contextPath, factoryClassName, classLoader, properties);
            } else {
                logger.log(Level.FINE, "Unable to load:{0}", resource.toString());
            }
        } catch (UnsupportedEncodingException e) {
            // should never happen
            throw new JAXBException(e);
        } catch (IOException e) {
            throw new JAXBException(e);
        } finally {
            try {
                if (r != null) {
                    r.close();
                }
            } catch (IOException ex) {
                Logger.getLogger(ContextFinder.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
```

```java

        // else no provider found
        logger.fine("Trying to create the platform default provider");
        return newInstance(contextPath, PLATFORM_DEFAULT_FACTORY_CLASS, classLoader, properties);
    }

    static JAXBContext find( Class[] classes, Map properties ) throws JAXBException {

        final String jaxbContextFQCN = JAXBContext.class.getName();
        String factoryClassName;

        // search for jaxb.properties in the class loader of each class first
        for (final Class c : classes) {
            // this classloader is used only to load jaxb.properties, so doing this should be safe.
            ClassLoader classLoader = getClassClassLoader(c);
            Package pkg = c.getPackage();
            if(pkg==null)
                continue;       // this is possible for primitives, arrays, and classes that are loaded by poorly implemented ClassLoaders
            String packageName = pkg.getName().replace('.', '/');

            // TODO: do we want to optimize away searching the same package?  org.Foo, org.Bar, com.Baz
            //       classes from the same package might come from different class loades, so it might be a bad idea

            // TODO: it's easier to look things up from the class
            // c.getResourceAsStream("jaxb.properties");

            // build the resource name and use the property loader code
            String resourceName = packageName+"/jaxb.properties";
            logger.log(Level.FINE, "Trying to locate {0}", resourceName);
            Properties props = loadJAXBProperties(classLoader, resourceName);
            if (props == null) {
                logger.fine("  not found");
            } else {
                logger.fine("  found");
                if (props.containsKey(JAXB_CONTEXT_FACTORY)) {
                    // trim() seems redundant, but adding to satisfy customer complaint
                    factoryClassName = props.getProperty(JAXB_CONTEXT_FACTORY).trim();
                    return newInstance(classes, properties, factoryClassName);
                } else {
                    throw new JAXBException(Messages.format(Messages.MISSING_PROPERTY, packageName, JAXB_CONTEXT_FACTORY));
                }
            }
        }

        // search for a system property second (javax.xml.bind.JAXBContext)
        logger.log(Level.FINE, "Checking system property {0}", JAXBContext.JAXB_CONTEXT_FACTORY);
        factoryClassName = AccessController.doPrivileged(new GetPropertyAction(JAXBContext.JAXB_CONTEXT_FACTORY));
        if (factoryClassName != null) {
```

```
            logger.log(Level.FINE, "  found {0}", factoryClassName);
            return newInstance( classes, properties, factoryClassName );
        } else { // leave it here for compatibility reasons
            logger.fine("  not found");
            logger.log(Level.FINE, "Checking system property {0}", jaxbContextFQCN);
            factoryClassName = AccessController.doPrivileged(new GetPropertyAction(jaxbContextFQCN)
                );
            if (factoryClassName != null) {
                logger.log(Level.FINE, "  found {0}", factoryClassName);
                return newInstance( classes, properties, factoryClassName );
            } else {
                logger.fine("  not found");
            }
        }

        // OSGi search
        Class jaxbContext = lookupJaxbContextUsingOsgiServiceLoader();
        if (jaxbContext != null) {
            logger.fine("OSGi environment detected");
            return newInstance(classes, properties, jaxbContext);
        }

        // search META-INF services next
        logger.fine("Checking META-INF/services");
        BufferedReader r = null;
        try {
            final String resource = new StringBuilder("META-INF/services/").append(jaxbContextFQCN)
                .toString();
            ClassLoader classLoader = getContextClassLoader();
            URL resourceURL;
            if(classLoader==null)
                resourceURL = ClassLoader.getSystemResource(resource);
            else
                resourceURL = classLoader.getResource(resource);

            if (resourceURL != null) {
                logger.log(Level.FINE, "Reading {0}", resourceURL);
                r = new BufferedReader(new InputStreamReader(resourceURL.openStream(), "UTF-8"));
                factoryClassName = r.readLine();
                if (factoryClassName != null) {
                    factoryClassName = factoryClassName.trim();
                }
                return newInstance(classes, properties, factoryClassName);
            } else {
                logger.log(Level.FINE, "Unable to find: {0}", resource);
            }
        } catch (UnsupportedEncodingException e) {
            // should never happen
            throw new JAXBException(e);
        } catch (IOException e) {
            throw new JAXBException(e);
        } finally {
            if (r != null) {
```

```
                try {
                    r.close();
                } catch (IOException ex) {
                    logger.log(Level.FINE, "Unable to close stream", ex);
                }
            }
        }

        // else no provider found
        logger.fine("Trying to create the platform default provider");
        return newInstance(classes, properties, PLATFORM_DEFAULT_FACTORY_CLASS);
    }

    private static Class lookupJaxbContextUsingOsgiServiceLoader() {
        try {
            // Use reflection to avoid having any dependency on ServiceLoader class
            Class target = Class.forName("com.sun.org.glassfish.hk2.osgiresourcelocator.
                ServiceLoader");
            Method m = target.getMethod("lookupProviderClasses", Class.class);
            Iterator iter = ((Iterable) m.invoke(null, JAXBContext.class)).iterator();
            return iter.hasNext() ? (Class)iter.next() : null;
        } catch(Exception e) {
            logger.log(Level.FINE, "Unable to find from OSGi: javax.xml.bind.JAXBContext");
            return null;
        }
    }

    private static Properties loadJAXBProperties( ClassLoader classLoader,
                                                  String propFileName )
        throws JAXBException {

        Properties props = null;

        try {
            URL url;
            if(classLoader==null)
                url = ClassLoader.getSystemResource(propFileName);
            else
                url = classLoader.getResource( propFileName );

            if( url != null ) {
                logger.log(Level.FINE, "loading props from {0}", url);
                props = new Properties();
                InputStream is = url.openStream();
                props.load( is );
                is.close();
            }
        } catch( IOException ioe ) {
            logger.log(Level.FINE,"Unable to load "+propFileName,ioe);
            throw new JAXBException( ioe.toString(), ioe );
        }

        return props;
```

```
    }


    /**
     * Search the given ClassLoader for an instance of the specified class and
     * return a string representation of the URL that points to the resource.
     *
     * @param clazz
     *          The class to search for
     * @param loader
     *          The ClassLoader to search.  If this parameter is null, then the
     *          system class loader will be searched
     * @return
     *          the URL for the class or null if it wasn't found
     */
    static URL which(Class clazz, ClassLoader loader) {

        String classnameAsResource = clazz.getName().replace('.', '/') + ".class";

        if(loader == null) {
            loader = getSystemClassLoader();
        }

        return loader.getResource(classnameAsResource);
    }

    /**
     * Get the URL for the Class from it's ClassLoader.
     *
     * Convenience method for {@link #which(Class, ClassLoader)}.
     *
     * Equivalent to calling: which(clazz, clazz.getClassLoader())
     *
     * @param clazz
     *          The class to search for
     * @return
     *          the URL for the class or null if it wasn't found
     */
    static URL which(Class clazz) {
        return which(clazz, getClassClassLoader(clazz));
    }

    /**
     * When JAXB is in J2SE, rt.jar has to have a JAXB implementation.
     * However, rt.jar cannot have META-INF/services/javax.xml.bind.JAXBContext
     * because if it has, it will take precedence over any file that applications have
     * in their jar files.
     *
     * <p>
     * When the user bundles his own JAXB implementation, we'd like to use it, and we
     * want the platform default to be used only when there's no other JAXB provider.
     *
     * <p>
```

```
     * For this reason, we have to hard-code the class name into the API.
     */
    private static final String PLATFORM_DEFAULT_FACTORY_CLASS = "com.sun.xml.internal.bind.v2.
        ContextFactory";

    /**
     * Loads the class, provided that the calling thread has an access to the class being loaded.
     */
    private static Class safeLoadClass(String className, ClassLoader classLoader) throws
        ClassNotFoundException {
       logger.log(Level.FINE, "Trying to load {0}", className);
       try {
          // make sure that the current thread has an access to the package of the given name.
          SecurityManager s = System.getSecurityManager();
          if (s != null) {
              int i = className.lastIndexOf('.');
              if (i != -1) {
                  s.checkPackageAccess(className.substring(0,i));
              }
          }

          if (classLoader == null) {
              return Class.forName(className);
          } else {
              return classLoader.loadClass(className);
          }
       } catch (SecurityException se) {
           // anyone can access the platform default factory class without permission
           if (PLATFORM_DEFAULT_FACTORY_CLASS.equals(className)) {
              return Class.forName(className);
           }
           throw se;
       }
    }

    private static ClassLoader getContextClassLoader() {
        if (System.getSecurityManager() == null) {
            return Thread.currentThread().getContextClassLoader();
        } else {
            return (ClassLoader) java.security.AccessController.doPrivileged(
                    new java.security.PrivilegedAction() {
                        public java.lang.Object run() {
                            return Thread.currentThread().getContextClassLoader();
                        }
                    });
        }
    }

    private static ClassLoader getClassClassLoader(final Class c) {
        if (System.getSecurityManager() == null) {
            return c.getClassLoader();
        } else {
            return (ClassLoader) java.security.AccessController.doPrivileged(
```

```
                    new java.security.PrivilegedAction() {
                        public java.lang.Object run() {
                            return c.getClassLoader();
                        }
                    });
        }
    }

    private static ClassLoader getSystemClassLoader() {
        if (System.getSecurityManager() == null) {
            return ClassLoader.getSystemClassLoader();
        } else {
            return (ClassLoader) java.security.AccessController.doPrivileged(
                    new java.security.PrivilegedAction() {
                        public java.lang.Object run() {
                            return ClassLoader.getSystemClassLoader();
                        }
                    });
        }
    }

}
```

2.3 DataBindingException.java

Secuencia 5: javax/xml/bind/DataBindingException.java

```
/*
 * Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

```

```
/**
 * Exception that represents a failure in a JAXB operation.
 *
 * <p>
 * This exception differs from {@link JAXBException} in that
 * this is an unchecked exception, while <tt>JAXBException</tt>
 * is a checked exception.
 *
 * @see JAXB
 * @since JAXB2.1
 */
public class DataBindingException extends RuntimeException {
    public DataBindingException(String message, Throwable cause) {
        super(message, cause);
    }

    public DataBindingException(Throwable cause) {
        super(cause);
    }
}
```

2.4 DatatypeConverter.java

Secuencia 6: javax/xml/bind/DatatypeConverter.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import javax.xml.namespace.NamespaceContext;

```

```
/**
 * <p>
 * The javaType binding declaration can be used to customize the binding of
 * an XML schema datatype to a Java datatype. Customizations can involve
 * writing a parse and print method for parsing and printing lexical
 * representations of a XML schema datatype respectively. However, writing
 * parse and print methods requires knowledge of the lexical representations (
 * <a href="http://www.w3.org/TR/xmlschema-2/"> XML Schema Part2: Datatypes
 * specification </a>) and hence may be difficult to write.
 * </p>
 * <p>
 * This class makes it easier to write parse and print methods. It defines
 * static parse and print methods that provide access to a JAXB provider's
 * implementation of parse and print methods. These methods are invoked by
 * custom parse and print methods. For example, the binding of xsd:dateTime
 * to a long can be customized using parse and print methods as follows:
 * <blockquote>
 *    <pre>
 *    // Customized parse method
 *    public long myParseCal( String dateTimeString ) {
 *        java.util.Calendar cal = DatatypeConverter.parseDateTime(dateTimeString);
 *        long longval = convert_calendar_to_long(cal); //application specific
 *        return longval;
 *    }
 *
 *    // Customized print method
 *    public String myPrintCal( Long longval ) {
 *        java.util.Calendar cal = convert_long_to_calendar(longval) ; //application specific
 *        String dateTimeString = DatatypeConverter.printDateTime(cal);
 *        return dateTimeString;
 *    }
 *    </pre>
 * </blockquote>
 * <p>
 * There is a static parse and print method corresponding to each parse and
 * print method respectively in the {@link DatatypeConverterInterface
 * DatatypeConverterInterface}.
 * <p>
 * The static methods defined in the class can also be used to specify
 * a parse or a print method in a javaType binding declaration.
 * </p>
 * <p>
 * JAXB Providers are required to call the
 * {@link #setDatatypeConverter(DatatypeConverterInterface)
 * setDatatypeConverter} api at some point before the first marshal or unmarshal
 * operation (perhaps during the call to JAXBContext.newInstance).  This step is
 * necessary to configure the converter that should be used to perform the
 * print and parse functionality.
 * </p>
 *
 * <p>
 * A print method for a XML schema datatype can output any lexical
 * representation that is valid with respect to the XML schema datatype.
```

```
 * If an error is encountered during conversion, then an IllegalArgumentException,
 * or a subclass of IllegalArgumentException must be thrown by the method.
 * </p>
 *
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems Inc
     .</li><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Ryan Shoemaker,Sun Microsystems
     Inc.</li></ul>
 * @see DatatypeConverterInterface
 * @see ParseConversionEvent
 * @see PrintConversionEvent
 * @since JAXB1.0
 */

final public class DatatypeConverter {

    // delegate to this instance of DatatypeConverter
    private static volatile DatatypeConverterInterface theConverter = null;

    private final static JAXBPermission SET_DATATYPE_CONVERTER_PERMISSION =
                            new JAXBPermission("setDatatypeConverter");

    private DatatypeConverter() {
        // private constructor
    }

    /**
     * This method is for JAXB provider use only.
     * <p>
     * JAXB Providers are required to call this method at some point before
     * allowing any of the JAXB client marshal or unmarshal operations to
     * occur.  This is necessary to configure the datatype converter that
     * should be used to perform the print and parse conversions.
     *
     * <p>
     * Calling this api repeatedly will have no effect - the
     * DatatypeConverterInterface instance passed into the first invocation is
     * the one that will be used from then on.
     *
     * @param converter an instance of a class that implements the
     * DatatypeConverterInterface class - this parameter must not be null.
     * @throws IllegalArgumentException if the parameter is null
     * @throws SecurityException
     *      If the {@link SecurityManager} in charge denies the access to
     *      set the datatype converter.
     * @see JAXBPermission
     */
    public static void setDatatypeConverter( DatatypeConverterInterface converter ) {
        if( converter == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.CONVERTER_MUST_NOT_BE_NULL ) );
        } else if( theConverter == null ) {
            SecurityManager sm = System.getSecurityManager();
            if (sm != null)
```

```
                sm.checkPermission(SET_DATATYPE_CONVERTER_PERMISSION);
            theConverter = converter;
        }
    }

    private static synchronized void initConverter() {
        theConverter = new DatatypeConverterImpl();
    }

    /**
     * <p>
     * Convert the lexical XSD string argument into a String value.
     * @param lexicalXSDString
     *     A string containing a lexical representation of
     *     xsd:string.
     * @return
     *     A String value represented by the string argument.
     */
    public static String parseString( String lexicalXSDString ) {
        if (theConverter == null) initConverter();
        return theConverter.parseString( lexicalXSDString );
    }

    /**
     * <p>
     * Convert the string argument into a BigInteger value.
     * @param lexicalXSDInteger
     *     A string containing a lexical representation of
     *     xsd:integer.
     * @return
     *     A BigInteger value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDInteger</code> is not a valid string
         representation of a {@link java.math.BigInteger} value.
     */
    public static java.math.BigInteger parseInteger( String lexicalXSDInteger ) {
        if (theConverter == null) initConverter();
        return theConverter.parseInteger( lexicalXSDInteger );
    }

    /**
     * <p>
     * Convert the string argument into an int value.
     * @param lexicalXSDInt
     *     A string containing a lexical representation of
     *     xsd:int.
     * @return
     *     A int value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDInt</code> is not a valid string
         representation of an <code>int</code> value.
     */
    public static int parseInt( String lexicalXSDInt ) {
        if (theConverter == null) initConverter();
        return theConverter.parseInt( lexicalXSDInt );
```

```
    }

    /**
     * <p>
     * Converts the string argument into a long value.
     * @param lexicalXSDLong
     *     A string containing lexical representation of
     *     xsd:long.
     * @return
     *     A long value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDLong</code> is not a valid string
         representation of a <code>long</code> value.
     */
    public static long parseLong( String lexicalXSDLong ) {
        if (theConverter == null) initConverter();
        return theConverter.parseLong( lexicalXSDLong );
    }

    /**
     * <p>
     * Converts the string argument into a short value.
     * @param lexicalXSDShort
     *     A string containing lexical representation of
     *     xsd:short.
     * @return
     *     A short value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDShort</code> is not a valid string
         representation of a <code>short</code> value.
     */
    public static short parseShort( String lexicalXSDShort ) {
        if (theConverter == null) initConverter();
        return theConverter.parseShort( lexicalXSDShort );
    }

    /**
     * <p>
     * Converts the string argument into a BigDecimal value.
     * @param lexicalXSDDecimal
     *     A string containing lexical representation of
     *     xsd:decimal.
     * @return
     *     A BigDecimal value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDDecimal</code> is not a valid string
         representation of {@link java.math.BigDecimal}.
     */
    public static java.math.BigDecimal parseDecimal( String lexicalXSDDecimal ) {
        if (theConverter == null) initConverter();
        return theConverter.parseDecimal( lexicalXSDDecimal );
    }

    /**
     * <p>
     * Converts the string argument into a float value.
```

```
     * @param lexicalXSDFloat
     *     A string containing lexical representation of
     *     xsd:float.
     * @return
     *     A float value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDFloat</code> is not a valid string
         representation of a <code>float</code> value.
     */
    public static float parseFloat( String lexicalXSDFloat ) {
        if (theConverter == null) initConverter();
        return theConverter.parseFloat( lexicalXSDFloat );
    }

    /**
     * <p>
     * Converts the string argument into a double value.
     * @param lexicalXSDDouble
     *     A string containing lexical representation of
     *     xsd:double.
     * @return
     *     A double value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDDouble</code> is not a valid string
         representation of a <code>double</code> value.
     */
    public static double parseDouble( String lexicalXSDDouble ) {
        if (theConverter == null) initConverter();
        return theConverter.parseDouble( lexicalXSDDouble );
    }

    /**
     * <p>
     * Converts the string argument into a boolean value.
     * @param lexicalXSDBoolean
     *     A string containing lexical representation of
     *     xsd:boolean.
     * @return
     *     A boolean value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
         defined in XML Schema Part 2: Datatypes for xsd:boolean.
     */
    public static boolean parseBoolean( String lexicalXSDBoolean ) {
        if (theConverter == null) initConverter();
        return theConverter.parseBoolean( lexicalXSDBoolean );
    }

    /**
     * <p>
     * Converts the string argument into a byte value.
     * @param lexicalXSDByte
     *     A string containing lexical representation of
     *     xsd:byte.
     * @return
     *     A byte value represented by the string argument.
```

```
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:byte.
     */
    public static byte parseByte( String lexicalXSDByte ) {
        if (theConverter == null) initConverter();
        return theConverter.parseByte( lexicalXSDByte );
    }

    /**
     * <p>
     * Converts the string argument into a byte value.
     *
     * <p>
     * String parameter <tt>lexicalXSDQname</tt> must conform to lexical value space specifed at
     * <a href="http://www.w3.org/TR/xmlschema-2/#QName">XML Schema Part 2:Datatypes specification:
         QNames</a>
     *
     * @param lexicalXSDQName
     *     A string containing lexical representation of xsd:QName.
     * @param nsc
     *     A namespace context for interpreting a prefix within a QName.
     * @return
     *     A QName value represented by the string argument.
     * @throws IllegalArgumentException  if string parameter does not conform to XML Schema Part 2
         specification or
     *      if namespace prefix of <tt>lexicalXSDQname</tt> is not bound to a URI in
         NamespaceContext <tt>nsc</tt>.
     */
    public static javax.xml.namespace.QName parseQName( String lexicalXSDQName,
                                                        NamespaceContext nsc) {
        if (theConverter == null) initConverter();
        return theConverter.parseQName( lexicalXSDQName, nsc );
    }

    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDDateTime
     *     A string containing lexical representation of
     *     xsd:datetime.
     * @return
     *     A Calendar object represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:dateTime.
     */
    public static java.util.Calendar parseDateTime( String lexicalXSDDateTime ) {
        if (theConverter == null) initConverter();
        return theConverter.parseDateTime( lexicalXSDDateTime );
    }

    /**
     * <p>
     * Converts the string argument into an array of bytes.
```

```
     * @param lexicalXSDBase64Binary
     *     A string containing lexical representation
     *     of xsd:base64Binary.
     * @return
     *     An array of bytes represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:base64Binary
     */
    public static byte[] parseBase64Binary( String lexicalXSDBase64Binary ) {
        if (theConverter == null) initConverter();
        return theConverter.parseBase64Binary( lexicalXSDBase64Binary );
    }

    /**
     * <p>
     * Converts the string argument into an array of bytes.
     * @param lexicalXSDHexBinary
     *     A string containing lexical representation of
     *     xsd:hexBinary.
     * @return
     *     An array of bytes represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:hexBinary.
     */
   public static byte[] parseHexBinary( String lexicalXSDHexBinary ) {
        if (theConverter == null) initConverter();
        return theConverter.parseHexBinary( lexicalXSDHexBinary );
    }

    /**
     * <p>
     * Converts the string argument into a long value.
     * @param lexicalXSDUnsignedInt
     *     A string containing lexical representation
     *     of xsd:unsignedInt.
     * @return
     *     A long value represented by the string argument.
     * @throws NumberFormatException if string parameter can not be parsed into a <tt>long</tt>
         value.
     */
    public static long parseUnsignedInt( String lexicalXSDUnsignedInt ) {
        if (theConverter == null) initConverter();
        return theConverter.parseUnsignedInt( lexicalXSDUnsignedInt );
    }

    /**
     * <p>
     * Converts the string argument into an int value.
     * @param lexicalXSDUnsignedShort
     *     A string containing lexical
     *     representation of xsd:unsignedShort.
     * @return
     *     An int value represented by the string argument.
```

```
     * @throws NumberFormatException if string parameter can not be parsed into an <tt>int</tt>
         value.
     */
    public static int   parseUnsignedShort( String lexicalXSDUnsignedShort ) {
        if (theConverter == null) initConverter();
        return theConverter.parseUnsignedShort( lexicalXSDUnsignedShort );
    }

    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDTime
     *     A string containing lexical representation of
     *     xsd:time.
     * @return
     *     A Calendar value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
         defined in XML Schema Part 2: Datatypes for xsd:Time.
     */
    public static java.util.Calendar parseTime( String lexicalXSDTime ) {
        if (theConverter == null) initConverter();
        return theConverter.parseTime( lexicalXSDTime );
    }
    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDDate
     *      A string containing lexical representation of
     *     xsd:Date.
     * @return
     *     A Calendar value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
         defined in XML Schema Part 2: Datatypes for xsd:Date.
     */
    public static java.util.Calendar parseDate( String lexicalXSDDate ) {
        if (theConverter == null) initConverter();
        return theConverter.parseDate( lexicalXSDDate );
    }

    /**
     * <p>
     * Return a string containing the lexical representation of the
     * simple type.
     * @param lexicalXSDAnySimpleType
     *     A string containing lexical
     *     representation of the simple type.
     * @return
     *     A string containing the lexical representation of the
     *     simple type.
     */
    public static String parseAnySimpleType( String lexicalXSDAnySimpleType ) {
        if (theConverter == null) initConverter();
        return theConverter.parseAnySimpleType( lexicalXSDAnySimpleType );
```

```
    }
    /**
     * <p>
     * Converts the string argument into a string.
     * @param val
     *     A string value.
     * @return
     *     A string containing a lexical representation of xsd:string.
     */
     // also indicate the print methods produce a lexical
     // representation for given Java datatypes.

    public static String printString( String val ) {
        if (theConverter == null) initConverter();
        return theConverter.printString( val );
    }

    /**
     * <p>
     * Converts a BigInteger value into a string.
     * @param val
     *     A BigInteger value
     * @return
     *     A string containing a lexical representation of xsd:integer
     * @throws IllegalArgumentException <tt>val</tt> is null.
     */
    public static String printInteger( java.math.BigInteger val ) {
        if (theConverter == null) initConverter();
        return theConverter.printInteger( val );
    }

    /**
     * <p>
     * Converts an int value into a string.
     * @param val
     *     An int value
     * @return
     *     A string containing a lexical representation of xsd:int
     */
    public static String printInt( int val ) {
        if (theConverter == null) initConverter();
        return theConverter.printInt( val );
    }

    /**
     * <p>
     * Converts A long value into a string.
     * @param val
     *     A long value
     * @return
     *     A string containing a lexical representation of xsd:long
     */
    public static String printLong( long val ) {
```

```
        if (theConverter == null) initConverter();
        return theConverter.printLong( val );
    }

    /**
     * <p>
     * Converts a short value into a string.
     * @param val
     *     A short value
     * @return
     *     A string containing a lexical representation of xsd:short
     */
    public static String printShort( short val ) {
        if (theConverter == null) initConverter();
        return theConverter.printShort( val );
    }

    /**
     * <p>
     * Converts a BigDecimal value into a string.
     * @param val
     *     A BigDecimal value
     * @return
     *     A string containing a lexical representation of xsd:decimal
     * @throws IllegalArgumentException <tt>val</tt> is null.
     */
    public static String printDecimal( java.math.BigDecimal val ) {
        if (theConverter == null) initConverter();
        return theConverter.printDecimal( val );
    }

    /**
     * <p>
     * Converts a float value into a string.
     * @param val
     *     A float value
     * @return
     *     A string containing a lexical representation of xsd:float
     */
    public static String printFloat( float val ) {
        if (theConverter == null) initConverter();
        return theConverter.printFloat( val );
    }

    /**
     * <p>
     * Converts a double value into a string.
     * @param val
     *     A double value
     * @return
     *     A string containing a lexical representation of xsd:double
     */
    public static String printDouble( double val ) {
```

```
        if (theConverter == null) initConverter();
        return theConverter.printDouble( val );
    }

    /**
     * <p>
     * Converts a boolean value into a string.
     * @param val
     *     A boolean value
     * @return
     *     A string containing a lexical representation of xsd:boolean
     */
    public static String printBoolean( boolean val ) {
        if (theConverter == null) initConverter();
        return theConverter.printBoolean( val );
    }

    /**
     * <p>
     * Converts a byte value into a string.
     * @param val
     *     A byte value
     * @return
     *     A string containing a lexical representation of xsd:byte
     */
    public static String printByte( byte val ) {
        if (theConverter == null) initConverter();
        return theConverter.printByte( val );
    }

    /**
     * <p>
     * Converts a QName instance into a string.
     * @param val
     *     A QName value
     * @param nsc
     *     A namespace context for interpreting a prefix within a QName.
     * @return
     *     A string containing a lexical representation of QName
     * @throws IllegalArgumentException if <tt>val</tt> is null or
     * if <tt>nsc</tt> is non-null or <tt>nsc.getPrefix(nsprefixFromVal)</tt> is null.
     */
    public static String printQName( javax.xml.namespace.QName val,
                                     NamespaceContext nsc ) {
        if (theConverter == null) initConverter();
        return theConverter.printQName( val, nsc );
    }

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
```

```
     * @return
     *     A string containing a lexical representation of xsd:dateTime
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public static String printDateTime( java.util.Calendar val ) {
        if (theConverter == null) initConverter();
        return theConverter.printDateTime( val );
    }

    /**
     * <p>
     * Converts an array of bytes into a string.
     * @param val
     *     An array of bytes
     * @return
     *     A string containing a lexical representation of xsd:base64Binary
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public static String printBase64Binary( byte[] val ) {
        if (theConverter == null) initConverter();
        return theConverter.printBase64Binary( val );
    }

    /**
     * <p>
     * Converts an array of bytes into a string.
     * @param val
     *     An array of bytes
     * @return
     *     A string containing a lexical representation of xsd:hexBinary
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public static String printHexBinary( byte[] val ) {
        if (theConverter == null) initConverter();
        return theConverter.printHexBinary( val );
    }

    /**
     * <p>
     * Converts a long value into a string.
     * @param val
     *     A long value
     * @return
     *     A string containing a lexical representation of xsd:unsignedInt
     */
    public static String printUnsignedInt( long val ) {
        if (theConverter == null) initConverter();
        return theConverter.printUnsignedInt( val );
    }

    /**
     * <p>
     * Converts an int value into a string.
```

```
     * @param val
     *     An int value
     * @return
     *     A string containing a lexical representation of xsd:unsignedShort
     */
    public static String printUnsignedShort( int val ) {
        if (theConverter == null) initConverter();
        return theConverter.printUnsignedShort( val );
    }

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
     * @return
     *     A string containing a lexical representation of xsd:time
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public static String printTime( java.util.Calendar val ) {
        if (theConverter == null) initConverter();
        return theConverter.printTime( val );
    }

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
     * @return
     *     A string containing a lexical representation of xsd:date
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public static String printDate( java.util.Calendar val ) {
        if (theConverter == null) initConverter();
        return theConverter.printDate( val );
    }

    /**
     * <p>
     * Converts a string value into a string.
     * @param val
     *     A string value
     * @return
     *     A string containing a lexical representation of xsd:AnySimpleType
     */
    public static String printAnySimpleType( String val ) {
        if (theConverter == null) initConverter();
        return theConverter.printAnySimpleType( val );
    }
}
```

2.5 DatatypeConverterImpl.java

Secuencia 7: javax/xml/bind/DatatypeConverterImpl.java

```
/*
 * Copyright (c) 2007, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.TimeZone;

import javax.xml.namespace.QName;
import javax.xml.namespace.NamespaceContext;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.DatatypeConfigurationException;

/**
 * This class is the JAXB RI's default implementation of the
 * {@link DatatypeConverterInterface}.
 *
 * <p>
 * When client applications specify the use of the static print/parse
 * methods in {@link DatatypeConverter}, it will delegate
 * to this class.
 *
 * <p>
 * This class is responsible for whitespace normalization.
 *
```

```
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @since JAXB2.1
 */
final class DatatypeConverterImpl implements DatatypeConverterInterface {

    /**
     * To avoid re-creating instances, we cache one instance.
     */
    public static final DatatypeConverterInterface theInstance = new DatatypeConverterImpl();

    protected DatatypeConverterImpl() {
    }

    public String parseString(String lexicalXSDString) {
        return lexicalXSDString;
    }

    public BigInteger parseInteger(String lexicalXSDInteger) {
        return _parseInteger(lexicalXSDInteger);
    }

    public static BigInteger _parseInteger(CharSequence s) {
        return new BigInteger(removeOptionalPlus(WhiteSpaceProcessor.trim(s)).toString());
    }

    public String printInteger(BigInteger val) {
        return _printInteger(val);
    }

    public static String _printInteger(BigInteger val) {
        return val.toString();
    }

    public int parseInt(String s) {
        return _parseInt(s);
    }

    /**
     * Faster but less robust String->int conversion.
     *
     * Note that:
     * <ol>
     *  <li>XML Schema allows '+', but {@link Integer#valueOf(String)} is not.
     *  <li>XML Schema allows leading and trailing (but not in-between) whitespaces.
     *      {@link Integer#valueOf(String)} doesn't allow any.
     * </ol>
     */
    public static int _parseInt(CharSequence s) {
        int len = s.length();
        int sign = 1;

        int r = 0;

```

```
        for (int i = 0; i < len; i++) {
            char ch = s.charAt(i);
            if (WhiteSpaceProcessor.isWhiteSpace(ch)) {
                // skip whitespace
            } else if ('0' <= ch && ch <= '9') {
                r = r * 10 + (ch - '0');
            } else if (ch == '-') {
                sign = -1;
            } else if (ch == '+') {
                // noop
            } else {
                throw new NumberFormatException("Not a number: " + s);
            }
        }

        return r * sign;
    }

    public long parseLong(String lexicalXSLong) {
        return _parseLong(lexicalXSLong);
    }

    public static long _parseLong(CharSequence s) {
        return Long.valueOf(removeOptionalPlus(WhiteSpaceProcessor.trim(s)).toString());
    }

    public short parseShort(String lexicalXSDShort) {
        return _parseShort(lexicalXSDShort);
    }

    public static short _parseShort(CharSequence s) {
        return (short) _parseInt(s);
    }

    public String printShort(short val) {
        return _printShort(val);
    }

    public static String _printShort(short val) {
        return String.valueOf(val);
    }

    public BigDecimal parseDecimal(String content) {
        return _parseDecimal(content);
    }

    public static BigDecimal _parseDecimal(CharSequence content) {
        content = WhiteSpaceProcessor.trim(content);

        if (content.length() <= 0) {
            return null;
        }

```

```
        return new BigDecimal(content.toString());

        // from purely XML Schema perspective,
        // this implementation has a problem, since
        // in xs:decimal "1.0" and "1" is equal whereas the above
        // code will return different values for those two forms.
        //
        // the code was originally using com.sun.msv.datatype.xsd.NumberType.load,
        // but a profiling showed that the process of normalizing "1.0" into "1"
        // could take non-trivial time.
        //
        // also, from the user's point of view, one might be surprised if
        // 1 (not 1.0) is returned from "1.000"
    }

    public float parseFloat(String lexicalXSDFloat) {
        return _parseFloat(lexicalXSDFloat);
    }

    public static float _parseFloat(CharSequence _val) {
        String s = WhiteSpaceProcessor.trim(_val).toString();
        /* Incompatibilities of XML Schema's float "xfloat" and Java's float "jfloat"

         * jfloat.valueOf ignores leading and trailing whitespaces,
        whereas this is not allowed in xfloat.
         * jfloat.valueOf allows "float type suffix" (f, F) to be
        appended after float literal (e.g., 1.52e-2f), whereare
        this is not the case of xfloat.

        gray zone
        ---------
         * jfloat allows ".523". And there is no clear statement that mentions
        this case in xfloat. Although probably this is allowed.
         *
         */

        if (s.equals("NaN")) {
            return Float.NaN;
        }
        if (s.equals("INF")) {
            return Float.POSITIVE_INFINITY;
        }
        if (s.equals("-INF")) {
            return Float.NEGATIVE_INFINITY;
        }

        if (s.length() == 0
                || !isDigitOrPeriodOrSign(s.charAt(0))
                || !isDigitOrPeriodOrSign(s.charAt(s.length() - 1))) {
            throw new NumberFormatException();
        }

        // these screening process is necessary due to the wobble of Float.valueOf method
```

```
        return Float.parseFloat(s);
    }

    public String printFloat(float v) {
        return _printFloat(v);
    }

    public static String _printFloat(float v) {
        if (Float.isNaN(v)) {
            return "NaN";
        }
        if (v == Float.POSITIVE_INFINITY) {
            return "INF";
        }
        if (v == Float.NEGATIVE_INFINITY) {
            return "-INF";
        }
        return String.valueOf(v);
    }

    public double parseDouble(String lexicalXSDDouble) {
        return _parseDouble(lexicalXSDDouble);
    }

    public static double _parseDouble(CharSequence _val) {
        String val = WhiteSpaceProcessor.trim(_val).toString();

        if (val.equals("NaN")) {
            return Double.NaN;
        }
        if (val.equals("INF")) {
            return Double.POSITIVE_INFINITY;
        }
        if (val.equals("-INF")) {
            return Double.NEGATIVE_INFINITY;
        }

        if (val.length() == 0
                || !isDigitOrPeriodOrSign(val.charAt(0))
                || !isDigitOrPeriodOrSign(val.charAt(val.length() - 1))) {
            throw new NumberFormatException(val);
        }


        // these screening process is necessary due to the wobble of Float.valueOf method
        return Double.parseDouble(val);
    }

    public boolean parseBoolean(String lexicalXSDBoolean) {
        Boolean b = _parseBoolean(lexicalXSDBoolean);
        return (b == null) ? false : b.booleanValue();
    }

```

```java
    public static Boolean _parseBoolean(CharSequence literal) {
        if (literal == null) {
            return null;
        }

        int i = 0;
        int len = literal.length();
        char ch;
        boolean value = false;

        if (literal.length() <= 0) {
            return null;
        }

        do {
            ch = literal.charAt(i++);
        } while (WhiteSpaceProcessor.isWhiteSpace(ch) && i < len);

        int strIndex = 0;

        switch (ch) {
            case '1':
                value = true;
                break;
            case '0':
                value = false;
                break;
            case 't':
                String strTrue = "rue";
                do {
                    ch = literal.charAt(i++);
                } while ((strTrue.charAt(strIndex++) == ch) && i < len && strIndex < 3);

                if (strIndex == 3) {
                    value = true;
                } else {
                    return false;
                }
//                    throw new IllegalArgumentException("String \"" + literal + "\" is not valid boolean value.");

                break;
            case 'f':
                String strFalse = "alse";
                do {
                    ch = literal.charAt(i++);
                } while ((strFalse.charAt(strIndex++) == ch) && i < len && strIndex < 4);


                if (strIndex == 4) {
                    value = false;
                } else {
                    return false;
```

```
                }
//                    throw new IllegalArgumentException("String \"" + literal + "\" is not valid boolean value.");

                break;
        }

        if (i < len) {
            do {
                ch = literal.charAt(i++);
            } while (WhiteSpaceProcessor.isWhiteSpace(ch) && i < len);
        }

        if (i == len) {
            return value;
        } else {
            return null;
        }
//            throw new IllegalArgumentException("String \"" + literal + "\" is not valid boolean value.");
    }

    public String printBoolean(boolean val) {
        return val ? "true" : "false";
    }

    public static String _printBoolean(boolean val) {
        return val ? "true" : "false";
    }

    public byte parseByte(String lexicalXSDByte) {
        return _parseByte(lexicalXSDByte);
    }

    public static byte _parseByte(CharSequence literal) {
        return (byte) _parseInt(literal);
    }

    public String printByte(byte val) {
        return _printByte(val);
    }

    public static String _printByte(byte val) {
        return String.valueOf(val);
    }

    public QName parseQName(String lexicalXSDQName, NamespaceContext nsc) {
        return _parseQName(lexicalXSDQName, nsc);
    }

    /**
     * @return null if fails to convert.
     */
```

```
    public static QName _parseQName(CharSequence text, NamespaceContext nsc) {
        int length = text.length();

        // trim whitespace
        int start = 0;
        while (start < length && WhiteSpaceProcessor.isWhiteSpace(text.charAt(start))) {
            start++;
        }

        int end = length;
        while (end > start && WhiteSpaceProcessor.isWhiteSpace(text.charAt(end - 1))) {
            end--;
        }

        if (end == start) {
            throw new IllegalArgumentException("input is empty");
        }


        String uri;
        String localPart;
        String prefix;

        // search ':'
        int idx = start + 1;    // no point in searching the first char. that's not valid.
        while (idx < end && text.charAt(idx) != ':') {
            idx++;
        }

        if (idx == end) {
            uri = nsc.getNamespaceURI("");
            localPart = text.subSequence(start, end).toString();
            prefix = "";
        } else {
            // Prefix exists, check everything
            prefix = text.subSequence(start, idx).toString();
            localPart = text.subSequence(idx + 1, end).toString();
            uri = nsc.getNamespaceURI(prefix);
            // uri can never be null according to javadoc,
            // but some users reported that there are implementations that return null.
            if (uri == null || uri.length() == 0) // crap. the NamespaceContext interface is broken
                .
            // error: unbound prefix
            {
                throw new IllegalArgumentException("prefix " + prefix + " is not bound to a
                    namespace");
            }
        }

        return new QName(uri, localPart, prefix);
    }

    public Calendar parseDateTime(String lexicalXSDDateTime) {
```

```
        return _parseDateTime(lexicalXSDDateTime);
    }

    public static GregorianCalendar _parseDateTime(CharSequence s) {
        String val = WhiteSpaceProcessor.trim(s).toString();
        return datatypeFactory.newXMLGregorianCalendar(val).toGregorianCalendar();
    }

    public String printDateTime(Calendar val) {
        return _printDateTime(val);
    }

    public static String _printDateTime(Calendar val) {
        return CalendarFormatter.doFormat("%Y-%M-%DT%h:%m:%s%z", val);
    }

    public byte[] parseBase64Binary(String lexicalXSDBase64Binary) {
        return _parseBase64Binary(lexicalXSDBase64Binary);
    }

    public byte[] parseHexBinary(String s) {
        final int len = s.length();

        // "111" is not a valid hex encoding.
        if (len % 2 != 0) {
            throw new IllegalArgumentException("hexBinary needs to be even-length: " + s);
        }

        byte[] out = new byte[len / 2];

        for (int i = 0; i < len; i += 2) {
            int h = hexToBin(s.charAt(i));
            int l = hexToBin(s.charAt(i + 1));
            if (h == -1 || l == -1) {
                throw new IllegalArgumentException("contains illegal character for hexBinary: " + s
                    );
            }

            out[i / 2] = (byte) (h * 16 + l);
        }

        return out;
    }

    private static int hexToBin(char ch) {
        if ('0' <= ch && ch <= '9') {
            return ch - '0';
        }
        if ('A' <= ch && ch <= 'F') {
            return ch - 'A' + 10;
        }
        if ('a' <= ch && ch <= 'f') {
            return ch - 'a' + 10;
```

```
        }
        return -1;
    }
    private static final char[] hexCode = "0123456789ABCDEF".toCharArray();

    public String printHexBinary(byte[] data) {
        StringBuilder r = new StringBuilder(data.length * 2);
        for (byte b : data) {
            r.append(hexCode[(b >> 4) & 0xF]);
            r.append(hexCode[(b & 0xF)]);
        }
        return r.toString();
    }

    public long parseUnsignedInt(String lexicalXSDUnsignedInt) {
        return _parseLong(lexicalXSDUnsignedInt);
    }

    public String printUnsignedInt(long val) {
        return _printLong(val);
    }

    public int parseUnsignedShort(String lexicalXSDUnsignedShort) {
        return _parseInt(lexicalXSDUnsignedShort);
    }

    public Calendar parseTime(String lexicalXSDTime) {
        return datatypeFactory.newXMLGregorianCalendar(lexicalXSDTime).toGregorianCalendar();
    }

    public String printTime(Calendar val) {
        return CalendarFormatter.doFormat("%h:%m:%s%z", val);
    }

    public Calendar parseDate(String lexicalXSDDate) {
        return datatypeFactory.newXMLGregorianCalendar(lexicalXSDDate).toGregorianCalendar();
    }

    public String printDate(Calendar val) {
        return _printDate(val);
    }

    public static String _printDate(Calendar val) {
        return CalendarFormatter.doFormat((new StringBuilder("%Y-%M-%D").append("%z")).toString(),
            val);
    }

    public String parseAnySimpleType(String lexicalXSDAnySimpleType) {
        return lexicalXSDAnySimpleType;
//        return (String)SimpleURType.theInstance._createValue( lexicalXSDAnySimpleType, null );
    }

    public String printString(String val) {
```

```
//        return StringType.theInstance.convertToLexicalValue( val, null );
        return val;
    }

    public String printInt(int val) {
        return _printInt(val);
    }

    public static String _printInt(int val) {
        return String.valueOf(val);
    }

    public String printLong(long val) {
        return _printLong(val);
    }

    public static String _printLong(long val) {
        return String.valueOf(val);
    }

    public String printDecimal(BigDecimal val) {
        return _printDecimal(val);
    }

    public static String _printDecimal(BigDecimal val) {
        return val.toPlainString();
    }

    public String printDouble(double v) {
        return _printDouble(v);
    }

    public static String _printDouble(double v) {
        if (Double.isNaN(v)) {
            return "NaN";
        }
        if (v == Double.POSITIVE_INFINITY) {
            return "INF";
        }
        if (v == Double.NEGATIVE_INFINITY) {
            return "-INF";
        }
        return String.valueOf(v);
    }

    public String printQName(QName val, NamespaceContext nsc) {
        return _printQName(val, nsc);
    }

    public static String _printQName(QName val, NamespaceContext nsc) {
        // Double-check
        String qname;
        String prefix = nsc.getPrefix(val.getNamespaceURI());
```

```
        String localPart = val.getLocalPart();

        if (prefix == null || prefix.length() == 0) { // be defensive
            qname = localPart;
        } else {
            qname = prefix + ':' + localPart;
        }

        return qname;
    }

    public String printBase64Binary(byte[] val) {
        return _printBase64Binary(val);
    }

    public String printUnsignedShort(int val) {
        return String.valueOf(val);
    }

    public String printAnySimpleType(String val) {
        return val;
    }

    /**
     * Just return the string passed as a parameter but
     * installs an instance of this class as the DatatypeConverter
     * implementation. Used from static fixed value initializers.
     */
    public static String installHook(String s) {
        DatatypeConverter.setDatatypeConverter(theInstance);
        return s;
    }
// base64 decoder
    private static final byte[] decodeMap = initDecodeMap();
    private static final byte PADDING = 127;

    private static byte[] initDecodeMap() {
        byte[] map = new byte[128];
        int i;
        for (i = 0; i < 128; i++) {
            map[i] = -1;
        }

        for (i = 'A'; i <= 'Z'; i++) {
            map[i] = (byte) (i - 'A');
        }
        for (i = 'a'; i <= 'z'; i++) {
            map[i] = (byte) (i - 'a' + 26);
        }
        for (i = '0'; i <= '9'; i++) {
            map[i] = (byte) (i - '0' + 52);
        }
        map['+'] = 62;
```

```java
        map['/'] = 63;
        map['='] = PADDING;

        return map;
    }

    /**
     * computes the length of binary data speculatively.
     *
     * <p>
     * Our requirement is to create byte[] of the exact length to store the binary data.
     * If we do this in a straight-forward way, it takes two passes over the data.
     * Experiments show that this is a non-trivial overhead (35% or so is spent on
     * the first pass in calculating the length.)
     *
     * <p>
     * So the approach here is that we compute the length speculatively, without looking
     * at the whole contents. The obtained speculative value is never less than the
     * actual length of the binary data, but it may be bigger. So if the speculation
     * goes wrong, we'll pay the cost of reallocation and buffer copying.
     *
     * <p>
     * If the base64 text is tightly packed with no indentation nor illegal char
     * (like what most web services produce), then the speculation of this method
     * will be correct, so we get the performance benefit.
     */
    private static int guessLength(String text) {
        final int len = text.length();

        // compute the tail '=' chars
        int j = len - 1;
        for (; j >= 0; j--) {
            byte code = decodeMap[text.charAt(j)];
            if (code == PADDING) {
                continue;
            }
            if (code == -1) // most likely this base64 text is indented. go with the upper bound
            {
                return text.length() / 4 * 3;
            }
            break;
        }

        j++;    // text.charAt(j) is now at some base64 char, so +1 to make it the size
        int padSize = len - j;
        if (padSize > 2) // something is wrong with base64. be safe and go with the upper bound
        {
            return text.length() / 4 * 3;
        }

        // so far this base64 looks like it's unindented tightly packed base64.
        // take a chance and create an array with the expected size
        return text.length() / 4 * 3 - padSize;
```

```java
    }

    /**
     * @param text
     *      base64Binary data is likely to be long, and decoding requires
     *      each character to be accessed twice (once for counting length, another
     *      for decoding.)
     *
     *      A benchmark showed that taking {@link String} is faster, presumably
     *      because JIT can inline a lot of string access (with data of 1K chars, it was twice as
         fast)
     */
    public static byte[] _parseBase64Binary(String text) {
        final int buflen = guessLength(text);
        final byte[] out = new byte[buflen];
        int o = 0;

        final int len = text.length();
        int i;

        final byte[] quadruplet = new byte[4];
        int q = 0;

        // convert each quadruplet to three bytes.
        for (i = 0; i < len; i++) {
            char ch = text.charAt(i);
            byte v = decodeMap[ch];

            if (v != -1) {
                quadruplet[q++] = v;
            }

            if (q == 4) {
                // quadruplet is now filled.
                out[o++] = (byte) ((quadruplet[0] << 2) | (quadruplet[1] >> 4));
                if (quadruplet[2] != PADDING) {
                    out[o++] = (byte) ((quadruplet[1] << 4) | (quadruplet[2] >> 2));
                }
                if (quadruplet[3] != PADDING) {
                    out[o++] = (byte) ((quadruplet[2] << 6) | (quadruplet[3]));
                }
                q = 0;
            }
        }

        if (buflen == o) // speculation worked out to be OK
        {
            return out;
        }

        // we overestimated, so need to create a new buffer
        byte[] nb = new byte[o];
        System.arraycopy(out, 0, nb, 0, o);
```

```
        return nb;
    }
    private static final char[] encodeMap = initEncodeMap();

    private static char[] initEncodeMap() {
        char[] map = new char[64];
        int i;
        for (i = 0; i < 26; i++) {
            map[i] = (char) ('A' + i);
        }
        for (i = 26; i < 52; i++) {
            map[i] = (char) ('a' + (i - 26));
        }
        for (i = 52; i < 62; i++) {
            map[i] = (char) ('0' + (i - 52));
        }
        map[62] = '+';
        map[63] = '/';

        return map;
    }

    public static char encode(int i) {
        return encodeMap[i & 0x3F];
    }

    public static byte encodeByte(int i) {
        return (byte) encodeMap[i & 0x3F];
    }

    public static String _printBase64Binary(byte[] input) {
        return _printBase64Binary(input, 0, input.length);
    }

    public static String _printBase64Binary(byte[] input, int offset, int len) {
        char[] buf = new char[((len + 2) / 3) * 4];
        int ptr = _printBase64Binary(input, offset, len, buf, 0);
        assert ptr == buf.length;
        return new String(buf);
    }

    /**
     * Encodes a byte array into a char array by doing base64 encoding.
     *
     * The caller must supply a big enough buffer.
     *
     * @return
     *      the value of {@code ptr+((len+2)/3)*4}, which is the new offset
     *      in the output buffer where the further bytes should be placed.
     */
    public static int _printBase64Binary(byte[] input, int offset, int len, char[] buf, int ptr) {
        // encode elements until only 1 or 2 elements are left to encode
        int remaining = len;
```

```java
        int i;
        for (i = offset;remaining >= 3; remaining -= 3, i += 3) {
            buf[ptr++] = encode(input[i] >> 2);
            buf[ptr++] = encode(
                    ((input[i] & 0x3) << 4)
                    | ((input[i + 1] >> 4) & 0xF));
            buf[ptr++] = encode(
                    ((input[i + 1] & 0xF) << 2)
                    | ((input[i + 2] >> 6) & 0x3));
            buf[ptr++] = encode(input[i + 2] & 0x3F);
        }
        // encode when exactly 1 element (left) to encode
        if (remaining == 1) {
            buf[ptr++] = encode(input[i] >> 2);
            buf[ptr++] = encode(((input[i]) & 0x3) << 4);
            buf[ptr++] = '=';
            buf[ptr++] = '=';
        }
        // encode when exactly 2 elements (left) to encode
        if (remaining == 2) {
            buf[ptr++] = encode(input[i] >> 2);
            buf[ptr++] = encode(((input[i] & 0x3) << 4)
                    | ((input[i + 1] >> 4) & 0xF));
            buf[ptr++] = encode((input[i + 1] & 0xF) << 2);
            buf[ptr++] = '=';
        }
        return ptr;
    }

    /**
     * Encodes a byte array into another byte array by first doing base64 encoding
     * then encoding the result in ASCII.
     *
     * The caller must supply a big enough buffer.
     *
     * @return
     *      the value of {@code ptr+((len+2)/3)*4}, which is the new offset
     *      in the output buffer where the further bytes should be placed.
     */
    public static int _printBase64Binary(byte[] input, int offset, int len, byte[] out, int ptr) {
        byte[] buf = out;
        int remaining = len;
        int i;
        for (i=offset; remaining >= 3; remaining -= 3, i += 3 ) {
            buf[ptr++] = encodeByte(input[i]>>2);
            buf[ptr++] = encodeByte(
                        ((input[i]&0x3)<<4) |
                        ((input[i+1]>>4)&0xF));
            buf[ptr++] = encodeByte(
                        ((input[i+1]&0xF)<<2)|
                        ((input[i+2]>>6)&0x3));
            buf[ptr++] = encodeByte(input[i+2]&0x3F);
        }
```

```
        // encode when exactly 1 element (left) to encode
        if (remaining == 1) {
            buf[ptr++] = encodeByte(input[i]>>2);
            buf[ptr++] = encodeByte(((input[i])&0x3)<<4);
            buf[ptr++] = '=';
            buf[ptr++] = '=';
        }
        // encode when exactly 2 elements (left) to encode
        if (remaining == 2) {
            buf[ptr++] = encodeByte(input[i]>>2);
            buf[ptr++] = encodeByte(
                        ((input[i]&0x3)<<4) |
                        ((input[i+1]>>4)&0xF));
            buf[ptr++] = encodeByte((input[i+1]&0xF)<<2);
            buf[ptr++] = '=';
        }

        return ptr;
    }

    private static CharSequence removeOptionalPlus(CharSequence s) {
        int len = s.length();

        if (len <= 1 || s.charAt(0) != '+') {
            return s;
        }

        s = s.subSequence(1, len);
        char ch = s.charAt(0);
        if ('0' <= ch && ch <= '9') {
            return s;
        }
        if ('.' == ch) {
            return s;
        }

        throw new NumberFormatException();
    }

    private static boolean isDigitOrPeriodOrSign(char ch) {
        if ('0' <= ch && ch <= '9') {
            return true;
        }
        if (ch == '+' || ch == '-' || ch == '.') {
            return true;
        }
        return false;
    }
    private static final DatatypeFactory datatypeFactory;

    static {
        try {
            datatypeFactory = DatatypeFactory.newInstance();
```

```
        } catch (DatatypeConfigurationException e) {
            throw new Error(e);
        }
    }

    private static final class CalendarFormatter {

        public static String doFormat(String format, Calendar cal) throws IllegalArgumentException
            {
            int fidx = 0;
            int flen = format.length();
            StringBuilder buf = new StringBuilder();

            while (fidx < flen) {
                char fch = format.charAt(fidx++);

                if (fch != '%') {  // not a meta character
                    buf.append(fch);
                    continue;
                }

                // seen meta character. we don't do error check against the format
                switch (format.charAt(fidx++)) {
                    case 'Y': // year
                        formatYear(cal, buf);
                        break;

                    case 'M': // month
                        formatMonth(cal, buf);
                        break;

                    case 'D': // days
                        formatDays(cal, buf);
                        break;

                    case 'h': // hours
                        formatHours(cal, buf);
                        break;

                    case 'm': // minutes
                        formatMinutes(cal, buf);
                        break;

                    case 's': // parse seconds.
                        formatSeconds(cal, buf);
                        break;

                    case 'z': // time zone
                        formatTimeZone(cal, buf);
                        break;

                    default:
                        // illegal meta character. impossible.
```

```
                    throw new InternalError();
            }
        }

        return buf.toString();
    }

    private static void formatYear(Calendar cal, StringBuilder buf) {
        int year = cal.get(Calendar.YEAR);

        String s;
        if (year <= 0) // negative value
        {
            s = Integer.toString(1 - year);
        } else // positive value
        {
            s = Integer.toString(year);
        }

        while (s.length() < 4) {
            s = '0' + s;
        }
        if (year <= 0) {
            s = '-' + s;
        }

        buf.append(s);
    }

    private static void formatMonth(Calendar cal, StringBuilder buf) {
        formatTwoDigits(cal.get(Calendar.MONTH) + 1, buf);
    }

    private static void formatDays(Calendar cal, StringBuilder buf) {
        formatTwoDigits(cal.get(Calendar.DAY_OF_MONTH), buf);
    }

    private static void formatHours(Calendar cal, StringBuilder buf) {
        formatTwoDigits(cal.get(Calendar.HOUR_OF_DAY), buf);
    }

    private static void formatMinutes(Calendar cal, StringBuilder buf) {
        formatTwoDigits(cal.get(Calendar.MINUTE), buf);
    }

    private static void formatSeconds(Calendar cal, StringBuilder buf) {
        formatTwoDigits(cal.get(Calendar.SECOND), buf);
        if (cal.isSet(Calendar.MILLISECOND)) { // milliseconds
            int n = cal.get(Calendar.MILLISECOND);
            if (n != 0) {
                String ms = Integer.toString(n);
                while (ms.length() < 3) {
                    ms = '0' + ms; // left 0 paddings.
```

```
                    }
                    buf.append('.');
                    buf.append(ms);
                }
            }
        }

        /** formats time zone specifier. */
        private static void formatTimeZone(Calendar cal, StringBuilder buf) {
            TimeZone tz = cal.getTimeZone();

            if (tz == null) {
                return;
            }

            // otherwise print out normally.
            int offset = tz.getOffset(cal.getTime().getTime());

            if (offset == 0) {
                buf.append('Z');
                return;
            }

            if (offset >= 0) {
                buf.append('+');
            } else {
                buf.append('-');
                offset *= -1;
            }

            offset /= 60 * 1000; // offset is in milli-seconds

            formatTwoDigits(offset / 60, buf);
            buf.append(':');
            formatTwoDigits(offset % 60, buf);
        }

        /** formats Integer into two-character-wide string. */
        private static void formatTwoDigits(int n, StringBuilder buf) {
            // n is always non-negative.
            if (n < 10) {
                buf.append('0');
            }
            buf.append(n);
        }
    }
}
```

2.6 DatatypeConverterInterface.java

Secuencia 8: javax/xml/bind/DatatypeConverterInterface.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * <p>
 * The DatatypeConverterInterface is for JAXB provider use only. A
 * JAXB provider must supply a class that implements this interface.
 * JAXB Providers are required to call the
 * {@link DatatypeConverter#setDatatypeConverter(DatatypeConverterInterface)
 * DatatypeConverter.setDatatypeConverter} api at
 * some point before the first marshal or unmarshal operation (perhaps during
 * the call to JAXBContext.newInstance).  This step is necessary to configure
 * the converter that should be used to perform the print and parse
 * functionality.  Calling this api repeatedly will have no effect - the
 * DatatypeConverter instance passed into the first invocation is the one that
 * will be used from then on.
 * </p>
 *
 * <p>
 * This interface defines the parse and print methods. There is one
 * parse and print method for each XML schema datatype specified in the
 * the default binding Table 5-1 in the JAXB specification.
 * </p>
 *
 * <p>
 * The parse and print methods defined here are invoked by the static parse
 * and print methods defined in the {@link DatatypeConverter DatatypeConverter}
 * class.
 * </p>
 *
 * <p>
```

```
 * A parse method for a XML schema datatype must be capable of converting any
 * lexical representation of the XML schema datatype ( specified by the
 * <a href="http://www.w3.org/TR/xmlschema-2/"> XML Schema Part2: Datatypes
 * specification</a> into a value in the value space of the XML schema datatype.
 * If an error is encountered during conversion, then an IllegalArgumentException
 * or a subclass of IllegalArgumentException must be thrown by the method.
 *
 * </p>
 *
 * <p>
 * A print method for a XML schema datatype can output any lexical
 * representation that is valid with respect to the XML schema datatype.
 * If an error is encountered during conversion, then an IllegalArgumentException,
 * or a subclass of IllegalArgumentException must be thrown by the method.
 * </p>
 *
 * The prefix xsd: is used to refer to XML schema datatypes
 * <a href="http://www.w3.org/TR/xmlschema-2/"> XML Schema Part2: Datatypes
 * specification.</a>
 *
 * <p>
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems Inc
     .</li><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Ryan Shoemaker,Sun Microsystems
     Inc.</li></ul>
 * @see DatatypeConverter
 * @see ParseConversionEvent
 * @see PrintConversionEvent
 * @since JAXB1.0
 */

public interface DatatypeConverterInterface {
    /**
     * <p>
     * Convert the string argument into a string.
     * @param lexicalXSDString
     *     A lexical representation of the XML Schema datatype xsd:string
     * @return
     *     A string that is the same as the input string.
     */
    public String parseString( String lexicalXSDString );

    /**
     * <p>
     * Convert the string argument into a BigInteger value.
     * @param lexicalXSDInteger
     *     A string containing a lexical representation of
     *     xsd:integer.
     * @return
     *     A BigInteger value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDInteger</code> is not a valid string
         representation of a {@link java.math.BigInteger} value.
     */
    public java.math.BigInteger parseInteger( String lexicalXSDInteger );
```

```java

    /**
     * <p>
     * Convert the string argument into an int value.
     * @param lexicalXSDInt
     *     A string containing a lexical representation of
     *     xsd:int.
     * @return
     *     An int value represented byte the string argument.
     * @throws NumberFormatException <code>lexicalXSDInt</code> is not a valid string
         representation of an <code>int</code> value.
     */
    public int parseInt( String lexicalXSDInt );

    /**
     * <p>
     * Converts the string argument into a long value.
     * @param lexicalXSDLong
     *     A string containing lexical representation of
     *     xsd:long.
     * @return
     *     A long value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDLong</code> is not a valid string
         representation of a <code>long</code> value.
     */
    public long parseLong( String lexicalXSDLong );

    /**
     * <p>
     * Converts the string argument into a short value.
     * @param lexicalXSDShort
     *     A string containing lexical representation of
     *     xsd:short.
     * @return
     *     A short value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDShort</code> is not a valid string
         representation of a <code>short</code> value.
     */
    public short parseShort( String lexicalXSDShort );

    /**
     * <p>
     * Converts the string argument into a BigDecimal value.
     * @param lexicalXSDDecimal
     *     A string containing lexical representation of
     *     xsd:decimal.
     * @return
     *     A BigDecimal value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDDecimal</code> is not a valid string
         representation of {@link java.math.BigDecimal}.
     */
    public java.math.BigDecimal parseDecimal( String lexicalXSDDecimal );

```

```java
    /**
     * <p>
     * Converts the string argument into a float value.
     * @param lexicalXSDFloat
     *     A string containing lexical representation of
     *     xsd:float.
     * @return
     *     A float value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDFloat</code> is not a valid string
         representation of a <code>float</code> value.
     */
    public float parseFloat( String lexicalXSDFloat );

    /**
     * <p>
     * Converts the string argument into a double value.
     * @param lexicalXSDDouble
     *     A string containing lexical representation of
     *     xsd:double.
     * @return
     *     A double value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDDouble</code> is not a valid string
         representation of a <code>double</code> value.
     */
    public double parseDouble( String lexicalXSDDouble );

    /**
     * <p>
     * Converts the string argument into a boolean value.
     * @param lexicalXSDBoolean
     *     A string containing lexical representation of
     *     xsd:boolean.
     * @return
     *     A boolean value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:boolean.
     */
    public boolean parseBoolean( String lexicalXSDBoolean );

    /**
     * <p>
     * Converts the string argument into a byte value.
     * @param lexicalXSDByte
     *     A string containing lexical representation of
     *     xsd:byte.
     * @return
     *     A byte value represented by the string argument.
     * @throws NumberFormatException <code>lexicalXSDByte</code> does not contain a parseable byte.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:byte.
     */
    public byte parseByte( String lexicalXSDByte );

```

```java
    /**
     * <p>
     * Converts the string argument into a QName value.
     *
     * <p>
     * String parameter <tt>lexicalXSDQname</tt> must conform to lexical value space specifed at
     * <a href="http://www.w3.org/TR/xmlschema-2/#QName">XML Schema Part 2:Datatypes specification: QNames</a>
     *
     * @param lexicalXSDQName
     *     A string containing lexical representation of xsd:QName.
     * @param nsc
     *     A namespace context for interpreting a prefix within a QName.
     * @return
     *     A QName value represented by the string argument.
     * @throws IllegalArgumentException  if string parameter does not conform to XML Schema Part 2 specification or
     *      if namespace prefix of <tt>lexicalXSDQname</tt> is not bound to a URI in NamespaceContext <tt>nsc</tt>.
     */
    public javax.xml.namespace.QName parseQName( String lexicalXSDQName,
                                                 javax.xml.namespace.NamespaceContext nsc);

    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDDateTime
     *     A string containing lexical representation of
     *     xsd:datetime.
     * @return
     *     A Calendar object represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space defined in XML Schema Part 2: Datatypes for xsd:dateTime.
     */
    public java.util.Calendar parseDateTime( String lexicalXSDDateTime );

    /**
     * <p>
     * Converts the string argument into an array of bytes.
     * @param lexicalXSDBase64Binary
     *     A string containing lexical representation
     *     of xsd:base64Binary.
     * @return
     *     An array of bytes represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space defined in XML Schema Part 2: Datatypes for xsd:base64Binary
     */
    public byte[] parseBase64Binary( String lexicalXSDBase64Binary );

    /**
     * <p>
     * Converts the string argument into an array of bytes.
     * @param lexicalXSDHexBinary
```

```
     *     A string containing lexical representation of
     *     xsd:hexBinary.
     * @return
     *     An array of bytes represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:hexBinary.
     */
    public byte[] parseHexBinary( String lexicalXSDHexBinary );

    /**
     * <p>
     * Converts the string argument into a long value.
     * @param lexicalXSDUnsignedInt
     *     A string containing lexical representation
     *     of xsd:unsignedInt.
     * @return
     *     A long value represented by the string argument.
     * @throws NumberFormatException if string parameter can not be parsed into a <tt>long</tt>
         value.
     */
    public long parseUnsignedInt( String lexicalXSDUnsignedInt );

    /**
     * <p>
     * Converts the string argument into an int value.
     * @param lexicalXSDUnsignedShort
     *     A string containing lexical
     *     representation of xsd:unsignedShort.
     * @return
     *     An int value represented by the string argument.
     * @throws NumberFormatException if string parameter can not be parsed into an <tt>int</tt>
         value.
     */
    public int parseUnsignedShort( String lexicalXSDUnsignedShort );

    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDTime
     *     A string containing lexical representation of
     *     xsd:Time.
     * @return
     *     A Calendar value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:Time.
     */
    public java.util.Calendar parseTime( String lexicalXSDTime );

    /**
     * <p>
     * Converts the string argument into a Calendar value.
     * @param lexicalXSDDate
     *     A string containing lexical representation of
```

```
     *     xsd:Date.
     * @return
     *     A Calendar value represented by the string argument.
     * @throws IllegalArgumentException if string parameter does not conform to lexical value space
          defined in XML Schema Part 2: Datatypes for xsd:Date.
     */
    public java.util.Calendar parseDate( String lexicalXSDDate );

    /**
     * <p>
     * Return a string containing the lexical representation of the
     * simple type.
     * @param lexicalXSDAnySimpleType
     *     A string containing lexical
     *     representation of the simple type.
     * @return
     *     A string containing the lexical representation of the
     *     simple type.
     */
    public String parseAnySimpleType( String lexicalXSDAnySimpleType );

    /**
     * <p>
     * Converts the string argument into a string.
     * @param val
     *     A string value.
     * @return
     *     A string containing a lexical representation of xsd:string
     */
    public String printString( String val );

    /**
     * <p>
     * Converts a BigInteger value into a string.
     * @param val
     *     A BigInteger value
     * @return
     *     A string containing a lexical representation of xsd:integer
     * @throws IllegalArgumentException <tt>val</tt> is null.
     */
    public String printInteger( java.math.BigInteger val );

    /**
     * <p>
     * Converts an int value into a string.
     * @param val
     *     An int value
     * @return
     *     A string containing a lexical representation of xsd:int
     */
    public String printInt( int val );


```

```java
    /**
     * <p>
     * Converts a long value into a string.
     * @param val
     *     A long value
     * @return
     *     A string containing a lexical representation of xsd:long
     */
    public String printLong( long val );

    /**
     * <p>
     * Converts a short value into a string.
     * @param val
     *     A short value
     * @return
     *     A string containing a lexical representation of xsd:short
     */
    public String printShort( short val );

    /**
     * <p>
     * Converts a BigDecimal value into a string.
     * @param val
     *     A BigDecimal value
     * @return
     *     A string containing a lexical representation of xsd:decimal
     * @throws IllegalArgumentException <tt>val</tt> is null.
     */
    public String printDecimal( java.math.BigDecimal val );

    /**
     * <p>
     * Converts a float value into a string.
     * @param val
     *     A float value
     * @return
     *     A string containing a lexical representation of xsd:float
     */
    public String printFloat( float val );

    /**
     * <p>
     * Converts a double value into a string.
     * @param val
     *     A double value
     * @return
     *     A string containing a lexical representation of xsd:double
     */
    public String printDouble( double val );

    /**
     * <p>
```

```
     * Converts a boolean value into a string.
     * @param val
     *     A boolean value
     * @return
     *     A string containing a lexical representation of xsd:boolean
     */
    public String printBoolean( boolean val );

    /**
     * <p>
     * Converts a byte value into a string.
     * @param val
     *     A byte value
     * @return
     *     A string containing a lexical representation of xsd:byte
     */
    public String printByte( byte val );

    /**
     * <p>
     * Converts a QName instance into a string.
     * @param val
     *     A QName value
     * @param nsc
     *     A namespace context for interpreting a prefix within a QName.
     * @return
     *     A string containing a lexical representation of QName
     * @throws IllegalArgumentException if <tt>val</tt> is null or
     * if <tt>nsc</tt> is non-null or <tt>nsc.getPrefix(nsprefixFromVal)</tt> is null.
     */
    public String printQName( javax.xml.namespace.QName val,
                              javax.xml.namespace.NamespaceContext nsc );

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
     * @return
     *     A string containing a lexical representation of xsd:dateTime
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public String printDateTime( java.util.Calendar val );

    /**
     * <p>
     * Converts an array of bytes into a string.
     * @param val
     *     an array of bytes
     * @return
     *     A string containing a lexical representation of xsd:base64Binary
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
```

```
    public String printBase64Binary( byte[] val );

    /**
     * <p>
     * Converts an array of bytes into a string.
     * @param val
     *     an array of bytes
     * @return
     *     A string containing a lexical representation of xsd:hexBinary
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public String printHexBinary( byte[] val );

    /**
     * <p>
     * Converts a long value into a string.
     * @param val
     *     A long value
     * @return
     *     A string containing a lexical representation of xsd:unsignedInt
     */
    public String printUnsignedInt( long val );

    /**
     * <p>
     * Converts an int value into a string.
     * @param val
     *     An int value
     * @return
     *     A string containing a lexical representation of xsd:unsignedShort
     */
    public String printUnsignedShort( int val );

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
     * @return
     *     A string containing a lexical representation of xsd:time
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
    public String printTime( java.util.Calendar val );

    /**
     * <p>
     * Converts a Calendar value into a string.
     * @param val
     *     A Calendar value
     * @return
     *     A string containing a lexical representation of xsd:date
     * @throws IllegalArgumentException if <tt>val</tt> is null.
     */
```

```
    public String printDate( java.util.Calendar val );

    /**
     * <p>
     * Converts a string value into a string.
     * @param val
     *     A string value
     * @return
     *     A string containing a lexical representation of xsd:AnySimpleType
     */
    public String printAnySimpleType( String val );
}
```

2.7 Element.java

Secuencia 9: javax/xml/bind/Element.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This is an element marker interface.
 *
 * Under certain circumstances, it is necessary for the binding compiler to
 * generate derived java content classes that implement this interface.  In
 * those cases, client applications must supply element instances rather than
 * types of elements.  For more detail, see section 5.7 "Element Declaration"
 * and 5.7.1 "Bind to Java Element Interface" of the specification.
 *
```

```
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @since JAXB1.0
 */

public interface Element {
}
```

2.8 GetPropertyAction.java

Secuencia 10: javax/xml/bind/GetPropertyAction.java

```
/*
 * Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import java.security.PrivilegedAction;

/**
 * {@link PrivilegedAction} that gets the system property value.
 * @author Kohsuke Kawaguchi
 */
final class GetPropertyAction implements PrivilegedAction<String> {
    private final String propertyName;

    public GetPropertyAction(String propertyName) {
        this.propertyName = propertyName;
    }

    public String run() {
        return System.getProperty(propertyName);
```

```
    }
}
```

2.9 JAXB.java

Secuencia 11: javax/xml/bind/JAXB.java

```
/*
 * Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.namespace.QName;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import java.beans.Introspector;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.Reader;
import java.io.Writer;
import java.lang.ref.WeakReference;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLConnection;

```

```java
/**
 * Class that defines convenience methods for common, simple use of JAXB.
 *
 * <p>
 * Methods defined in this class are convenience methods that combine several basic operations
 * in the {@link JAXBContext}, {@link Unmarshaller}, and {@link Marshaller}.
 *
 * They are designed
 * to be the prefered methods for developers new to JAXB. They have
 * the following characterstics:
 *
 * <ol>
 *  <li>Generally speaking, the performance is not necessarily optimal.
 *      It is expected that people who need to write performance
 *      critical code will use the rest of the JAXB API directly.
 *  <li>Errors that happen during the processing is wrapped into
 *      {@link DataBindingException} (which will have {@link JAXBException}
 *      as its {@link Throwable#getCause() cause}. It is expected that
 *      people who prefer the checked exception would use
 *      the rest of the JAXB API directly.
 * </ol>
 *
 * <p>
 * In addition, the <tt>unmarshal</tt> methods have the following characteristic:
 *
 * <ol>
 *  <li>Schema validation is not performed on the input XML.
 *      The processing will try to continue even if there
 *      are errors in the XML, as much as possible. Only as
 *      the last resort, this method fails with {@link DataBindingException}.
 * </ol>
 *
 * <p>
 * Similarly, the <tt>marshal</tt> methods have the following characteristic:
 * <ol>
 *  <li>The processing will try to continue even if the Java object tree
 *      does not meet the validity requirement. Only as
 *      the last resort, this method fails with {@link DataBindingException}.
 * </ol>
 *
 *
 * <p>
 * All the methods on this class require non-null arguments to all parameters.
 * The <tt>unmarshal</tt> methods either fail with an exception or return
 * a non-null value.
 *
 * @author Kohsuke Kawaguchi
 * @since 2.1
 */
public final class JAXB {
    /**
     * No instanciation is allowed.
     */
```

```java
    private JAXB() {}

    /**
     * To improve the performance, we'll cache the last {@link JAXBContext} used.
     */
    private static final class Cache {
        final Class type;
        final JAXBContext context;

        public Cache(Class type) throws JAXBException {
            this.type = type;
            this.context = JAXBContext.newInstance(type);
        }
    }

    /**
     * Cache. We don't want to prevent the {@link Cache#type} from GC-ed,
     * hence {@link WeakReference}.
     */
    private static volatile WeakReference<Cache> cache;

    /**
     * Obtains the {@link JAXBContext} from the given type,
     * by using the cache if possible.
     *
     * <p>
     * We don't use locks to control access to {@link #cache}, but this code
     * should be thread-safe thanks to the immutable {@link Cache} and {@code volatile}.
     */
    private static <T> JAXBContext getContext(Class<T> type) throws JAXBException {
        WeakReference<Cache> c = cache;
        if(c!=null) {
            Cache d = c.get();
            if(d!=null && d.type==type)
                return d.context;
        }

        // overwrite the cache
        Cache d = new Cache(type);
        cache = new WeakReference<Cache>(d);

        return d.context;
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      Reads the entire file as XML.
     */
    public static <T> T unmarshal( File xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(new StreamSource(
```

```
                xml), type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      The resource pointed by the URL is read in its entirety.
     */
    public static <T> T unmarshal( URL xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      The URI is {@link URI#toURL() turned into URL} and then
     *      follows the handling of <tt>URL</tt>.
     */
    public static <T> T unmarshal( URI xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      The string is first interpreted as an absolute <tt>URI</tt>.
     *      If it's not {@link URI#isAbsolute() a valid absolute URI},
     *      then it's interpreted as a <tt>File</tt>
     */
    public static <T> T unmarshal( String xml, Class<T> type ) {
```

```java
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      The entire stream is read as an XML infoset.
     *      Upon a successful completion, the stream will be closed by this method.
     */
    public static <T> T unmarshal( InputStream xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
     * @param xml
     *      The character stream is read as an XML infoset.
     *      The encoding declaration in the XML will be ignored.
     *      Upon a successful completion, the stream will be closed by this method.
     */
    public static <T> T unmarshal( Reader xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    /**
     * Reads in a Java object tree from the given XML input.
     *
```

```
     * @param xml
     *      The XML infoset that the {@link Source} represents is read.
     */
    public static <T> T unmarshal( Source xml, Class<T> type ) {
        try {
            JAXBElement<T> item = getContext(type).createUnmarshaller().unmarshal(toSource(xml),
                type);
            return item.getValue();
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }




    /**
     * Creates {@link Source} from various XML representation.
     * See {@link #unmarshal} for the conversion rules.
     */
    private static Source toSource(Object xml) throws IOException {
        if(xml==null)
            throw new IllegalArgumentException("no XML is given");

        if (xml instanceof String) {
            try {
                xml=new URI((String)xml);
            } catch (URISyntaxException e) {
                xml=new File((String)xml);
            }
        }
        if (xml instanceof File) {
            File file = (File) xml;
            return new StreamSource(file);
        }
        if (xml instanceof URI) {
            URI uri = (URI) xml;
            xml=uri.toURL();
        }
        if (xml instanceof URL) {
            URL url = (URL) xml;
            return new StreamSource(url.toExternalForm());
        }
        if (xml instanceof InputStream) {
            InputStream in = (InputStream) xml;
            return new StreamSource(in);
        }
        if (xml instanceof Reader) {
            Reader r = (Reader) xml;
            return new StreamSource(r);
        }
        if (xml instanceof Source) {
```

```
            return (Source) xml;
        }
        throw new IllegalArgumentException("I don't understand how to handle "+xml.getClass());
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      XML will be written to this file. If it already exists,
     *      it will be overwritten.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, File xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The XML will be {@link URLConnection#getOutputStream() sent} to the
     *      resource pointed by this URL. Note that not all <tt>URL</tt>s support
     *      such operation, and exact semantics depends on the <tt>URL</tt>
     *      implementations. In case of {@link HttpURLConnection HTTP URLs},
     *      this will perform HTTP POST.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, URL xml ) {
```

```
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The URI is {@link URI#toURL() turned into URL} and then
     *      follows the handling of <tt>URL</tt>. See above.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, URI xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The string is first interpreted as an absolute <tt>URI</tt>.
     *      If it's not {@link URI#isAbsolute() a valid absolute URI},
     *      then it's interpreted as a <tt>File</tt>
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, String xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
```

```
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The XML will be sent to the given {@link OutputStream}.
     *      Upon a successful completion, the stream will be closed by this method.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, OutputStream xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The XML will be sent as a character stream to the given {@link Writer}.
     *      Upon a successful completion, the stream will be closed by this method.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, Writer xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
```

```
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      The XML will be sent to the {@link Result} object.
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    public static void marshal( Object jaxbObject, Result xml ) {
        _marshal(jaxbObject,xml);
    }

    /**
     * Writes a Java object tree to XML and store it to the specified location.
     *
     * <p>
     * This method is a convenience method that combines several basic operations
     * in the {@link JAXBContext} and {@link Marshaller}. This method is designed
     * to be the prefered method for developers new to JAXB. This method
     * has the following characterstics:
     *
     * <ol>
     *  <li>Generally speaking, the performance is not necessarily optimal.
     *      It is expected that those people who need to write performance
     *      critical code will use the rest of the JAXB API directly.
     *  <li>Errors that happen during the processing is wrapped into
     *      {@link DataBindingException} (which will have {@link JAXBException}
     *      as its {@link Throwable#getCause() cause}. It is expected that
     *      those people who prefer the checked exception would use
     *      the rest of the JAXB API directly.
     * </ol>
     *
     * @param jaxbObject
     *      The Java object to be marshalled into XML. If this object is
     *      a {@link JAXBElement}, it will provide the root tag name and
     *      the body. If this object has {@link XmlRootElement}
     *      on its class definition, that will be used as the root tag name
     *      and the given object will provide the body. Otherwise,
     *      the root tag name is {@link Introspector#decapitalize(String) infered} from
     *      {@link Class#getSimpleName() the short class name}.
     *      This parameter must not be null.
     *
     * @param xml
     *      Represents the receiver of XML. Objects of the following types are allowed.
     *
     *      <table><tr>
     *          <th>Type</th>
     *          <th>Operation</th>
```

```
     *      </tr><tr>
     *          <td>{@link File}</td>
     *          <td>XML will be written to this file. If it already exists,
     *              it will be overwritten.</td>
     *      </tr><tr>
     *          <td>{@link URL}</td>
     *          <td>The XML will be {@link URLConnection#getOutputStream() sent} to the
     *              resource pointed by this URL. Note that not all <tt>URL</tt>s support
     *              such operation, and exact semantics depends on the <tt>URL</tt>
     *              implementations. In case of {@link HttpURLConnection HTTP URLs},
     *              this will perform HTTP POST.</td>
     *      </tr><tr>
     *          <td>{@link URI}</td>
     *          <td>The URI is {@link URI#toURL() turned into URL} and then
     *              follows the handling of <tt>URL</tt>. See above.</td>
     *      </tr><tr>
     *          <td>{@link String}</td>
     *          <td>The string is first interpreted as an absolute <tt>URI</tt>.
     *              If it's not {@link URI#isAbsolute() a valid absolute URI},
     *              then it's interpreted as a <tt>File</tt></td>
     *      </tr><tr>
     *          <td>{@link OutputStream}</td>
     *          <td>The XML will be sent to the given {@link OutputStream}.
     *              Upon a successful completion, the stream will be closed by this method.</td>
     *      </tr><tr>
     *          <td>{@link Writer}</td>
     *          <td>The XML will be sent as a character stream to the given {@link Writer}.
     *              Upon a successful completion, the stream will be closed by this method.</td>
     *      </tr><tr>
     *          <td>{@link Result}</td>
     *          <td>The XML will be sent to the {@link Result} object.</td>
     *      </tr></table>
     *
     * @throws DataBindingException
     *      If the operation fails, such as due to I/O error, unbindable classes.
     */
    private static void _marshal( Object jaxbObject, Object xml ) {
        try {
            JAXBContext context;

            if(jaxbObject instanceof JAXBElement) {
                context = getContext(((JAXBElement<?>)jaxbObject).getDeclaredType());
            } else {
                Class<?> clazz = jaxbObject.getClass();
                XmlRootElement r = clazz.getAnnotation(XmlRootElement.class);
                context = getContext(clazz);
                if(r==null) {
                    // we need to infer the name
                    jaxbObject = new JAXBElement(new QName(inferName(clazz)),clazz,jaxbObject);
                }
            }

            Marshaller m = context.createMarshaller();
```

```
            m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,true);
            m.marshal(jaxbObject, toResult(xml));
        } catch (JAXBException e) {
            throw new DataBindingException(e);
        } catch (IOException e) {
            throw new DataBindingException(e);
        }
    }

    private static String inferName(Class clazz) {
        return Introspector.decapitalize(clazz.getSimpleName());
    }

    /**
     * Creates {@link Result} from various XML representation.
     * See {@link #_marshal(Object,Object)} for the conversion rules.
     */
    private static Result toResult(Object xml) throws IOException {
        if(xml==null)
            throw new IllegalArgumentException("no XML is given");

        if (xml instanceof String) {
            try {
                xml=new URI((String)xml);
            } catch (URISyntaxException e) {
                xml=new File((String)xml);
            }
        }
        if (xml instanceof File) {
            File file = (File) xml;
            return new StreamResult(file);
        }
        if (xml instanceof URI) {
            URI uri = (URI) xml;
            xml=uri.toURL();
        }
        if (xml instanceof URL) {
            URL url = (URL) xml;
            URLConnection con = url.openConnection();
            con.setDoOutput(true);
            con.setDoInput(false);
            con.connect();
            return new StreamResult(con.getOutputStream());
        }
        if (xml instanceof OutputStream) {
            OutputStream os = (OutputStream) xml;
            return new StreamResult(os);
        }
        if (xml instanceof Writer) {
            Writer w = (Writer)xml;
            return new StreamResult(w);
        }
        if (xml instanceof Result) {
```

```
            return (Result) xml;
        }
        throw new IllegalArgumentException("I don't understand how to handle "+xml.getClass());
    }

}
```

2.10 JAXBContext.java

Secuencia 12: javax/xml/bind/JAXBContext.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import org.w3c.dom.Node;

import java.util.Collections;
import java.util.Map;
import java.util.Properties;
import java.io.IOException;
import java.io.InputStream;

/**
 * <p>
 * The <tt>JAXBContext</tt> class provides the client's entry point to the
 * JAXB API. It provides an abstraction for managing the XML/Java binding
 * information necessary to implement the JAXB binding framework operations:
 * unmarshal, marshal and validate.
 *
 * <p>A client application normally obtains new instances of this class using
```

```
* one of these two styles for newInstance methods, although there are other
* specialized forms of the method available:
*
* <ul>
*   <li>{@link #newInstance(String,ClassLoader) JAXBContext.newInstance( "com.acme.foo:com.acme.bar" )} <br/>
*   The JAXBContext instance is initialized from a list of colon
*   separated Java package names. Each java package contains
*   JAXB mapped classes, schema-derived classes and/or user annotated
*   classes. Additionally, the java package may contain JAXB package annotations
*   that must be processed. (see JLS, Section 7.4.1 "Named Packages").
*   </li>
*   <li>{@link #newInstance(Class...) JAXBContext.newInstance( com.acme.foo.Foo.class )} <br/>
*    The JAXBContext instance is initialized with class(es)
*    passed as parameter(s) and classes that are statically reachable from
*    these class(es). See {@link #newInstance(Class...)} for details.
*   </li>
* </ul>
*
* <p>
* <i><B>SPEC REQUIREMENT:</B> the provider must supply an implementation
* class containing the following method signatures:</i>
*
* <pre>
* public static JAXBContext createContext( String contextPath, ClassLoader classLoader, Map&lt;String,Object> properties ) throws JAXBException
* public static JAXBContext createContext( Class[] classes, Map&lt;String,Object> properties ) throws JAXBException
* </pre>
*
* <p><i>
* The following JAXB 1.0 requirement is only required for schema to
* java interface/implementation binding. It does not apply to JAXB annotated
* classes. JAXB Providers must generate a <tt>jaxb.properties</tt> file in
* each package containing schema derived classes.  The property file must
* contain a property named <tt>javax.xml.bind.context.factory</tt> whose
* value is the name of the class that implements the <tt>createContext</tt>
* APIs.</i>
*
* <p><i>
* The class supplied by the provider does not have to be assignable to
* <tt>javax.xml.bind.JAXBContext</tt>, it simply has to provide a class that
* implements the <tt>createContext</tt> APIs.</i>
*
* <p><i>
* In addition, the provider must call the
* {@link DatatypeConverter#setDatatypeConverter(DatatypeConverterInterface)
* DatatypeConverter.setDatatypeConverter} api prior to any client
* invocations of the marshal and unmarshal methods.  This is necessary to
* configure the datatype converter that will be used during these operations.</i>
*
* <a name="Unmarshalling"></a>
* <h3>Unmarshalling</h3>
```

```
* <p>
* The {@link Unmarshaller} class provides the client application the ability
* to convert XML data into a tree of Java content objects.
* The unmarshal method allows for
* any global XML element declared in the schema to be unmarshalled as
* the root of an instance document.
* Additionally, the unmarshal method allows for an unrecognized root element that
* has  an xsi:type attribute's value that references a type definition declared in
* the schema  to be unmarshalled as the root of an instance document.
* The <tt>JAXBContext</tt> object
* allows the merging of global elements and type definitions across a set of schemas (listed
* in the <tt>contextPath</tt>). Since each schema in the schema set can belong
* to distinct namespaces, the unification of schemas to an unmarshalling
* context should be namespace independent.  This means that a client
* application is able to unmarshal XML documents that are instances of
* any of the schemas listed in the <tt>contextPath</tt>.  For example:
*
* <pre>
*        JAXBContext jc = JAXBContext.newInstance( "com.acme.foo:com.acme.bar" );
*        Unmarshaller u = jc.createUnmarshaller();
*        FooObject fooObj = (FooObject)u.unmarshal( new File( "foo.xml" ) ); // ok
*        BarObject barObj = (BarObject)u.unmarshal( new File( "bar.xml" ) ); // ok
*        BazObject bazObj = (BazObject)u.unmarshal( new File( "baz.xml" ) ); // error, "com.acme.
    baz" not in contextPath
* </pre>
*
* <p>
* The client application may also generate Java content trees explicitly rather
* than unmarshalling existing XML data.  For all JAXB-annotated value classes,
* an application can create content using constructors.
* For schema-derived interface/implementation classes and for the
* creation of elements that are not bound to a JAXB-annotated
* class, an application needs to have access and knowledge about each of
* the schema derived <tt> ObjectFactory</tt> classes that exist in each of
* java packages contained in the <tt>contextPath</tt>.  For each schema
* derived java class, there is a static factory method that produces objects
* of that type.  For example,
* assume that after compiling a schema, you have a package <tt>com.acme.foo</tt>
* that contains a schema derived interface named <tt>PurchaseOrder</tt>.  In
* order to create objects of that type, the client application would use the
* factory method like this:
*
* <pre>
*       com.acme.foo.PurchaseOrder po =
*           com.acme.foo.ObjectFactory.createPurchaseOrder();
* </pre>
*
* <p>
* Once the client application has an instance of the the schema derived object,
* it can use the mutator methods to set content on it.
*
* <p>
* For more information on the generated <tt>ObjectFactory</tt> classes, see
```

```
* Section 4.2 <i>Java Package</i> of the specification.
*
* <p>
* <i><B>SPEC REQUIREMENT:</B> the provider must generate a class in each
* package that contains all of the necessary object factory methods for that
* package named ObjectFactory as well as the static
* <tt>newInstance( javaContentInterface )</tt> method</i>
*
* <h3>Marshalling</h3>
* <p>
* The {@link Marshaller} class provides the client application the ability
* to convert a Java content tree back into XML data.  There is no difference
* between marshalling a content tree that is created manually using the factory
* methods and marshalling a content tree that is the result an <tt>unmarshal
* </tt> operation.  Clients can marshal a java content tree back to XML data
* to a <tt>java.io.OutputStream</tt> or a <tt>java.io.Writer</tt>.  The
* marshalling process can alternatively produce SAX2 event streams to a
* registered <tt>ContentHandler</tt> or produce a DOM Node object.
* Client applications have control over the output encoding as well as
* whether or not to marshal the XML data as a complete document or
* as a fragment.
*
* <p>
* Here is a simple example that unmarshals an XML document and then marshals
* it back out:
*
* <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*
*       // unmarshal from foo.xml
*       Unmarshaller u = jc.createUnmarshaller();
*       FooObject fooObj = (FooObject)u.unmarshal( new File( "foo.xml" ) );
*
*       // marshal to System.out
*       Marshaller m = jc.createMarshaller();
*       m.marshal( fooObj, System.out );
* </pre>
*
*
* <h3>Validation</h3>
* <p>
* Validation has been changed significantly since JAXB 1.0.  The {@link Validator}
* class has been deprecated and made optional.  This means that you are advised
* not to use this class and, in fact, it may not even be available depending on
* your JAXB provider.  JAXB 1.0 client applications that rely on <tt>Validator</tt>
* will still work properly when deployed with the JAXB 1.0 runtime system.
*
* In JAXB 2.0, the {@link Unmarshaller} has included convenince methods that expose
* the JAXP 1.3 {@link javax.xml.validation} framework.  Please refer to the
* {@link Unmarshaller#setSchema(javax.xml.validation.Schema)} API for more
* information.
*
*
```

```
* <h3>JAXB Runtime Binding Framework Compatibility</h3>
* <p>
* The following JAXB 1.0 restriction only applies to binding schema to
* interfaces/implementation classes.
* Since this binding does not require a common runtime system, a JAXB
* client application must not attempt to mix runtime objects (<tt>JAXBContext,
* Marshaller</tt>, etc. ) from different providers.  This does not
* mean that the client application isn't portable, it simply means that a
* client has to use a runtime system provided by the same provider that was
* used to compile the schema.
*
*
* <h3>Discovery of JAXB implementation</h3>
* <p>
* When one of the <tt>newInstance</tt> methods is called, a JAXB implementation is discovered
* by the following steps.
*
* <ol>
* <li>
* For each package/class explicitly passed in to the {@link #newInstance} method, in the order they are specified,
* <tt>jaxb.properties</tt> file is looked up in its package, by using the associated classloader &mdash;
* this is {@link Class#getClassLoader() the owner class loader} for a {@link Class} argument, and for a package
* the specified {@link ClassLoader}.
*
* <p>
* If such a file is discovered, it is {@link Properties#load(InputStream) loaded} as a property file, and
* the value of the {@link #JAXB_CONTEXT_FACTORY} key will be assumed to be the provider factory class.
* This class is then loaded by the associated classloader discussed above.
*
* <p>
* This phase of the look up allows some packages to force the use of a certain JAXB implementation.
* (For example, perhaps the schema compiler has generated some vendor extension in the code.)
*
* <li>
* If the system property {@link #JAXB_CONTEXT_FACTORY} exists, then its value is assumed to be the provider
* factory class. This phase of the look up enables per-JVM override of the JAXB implementation.
*
* <li>
* Look for <tt>/META-INF/services/javax.xml.bind.JAXBContext</tt> file in the associated classloader.
* This file follows the standard service descriptor convention, and if such a file exists, its content
* is assumed to be the provider factory class. This phase of the look up is for automatic discovery.
* It allows users to just put a JAXB implementation in a classpath and use it without any furhter configuration.
```

```
 *
 * <li>
 * Finally, if all the steps above fail, then the rest of the look up is unspecified. That said,
 * the recommended behavior is to simply look for some hard-coded platform default JAXB implementation.
 * This phase of the look up is so that JavaSE can have its own JAXB implementation as the last resort.
 * </ol>
 *
 * <p>
 * Once the provider factory class is discovered, its
 * <tt>public static JAXBContext createContext(String,ClassLoader,Map)</tt> method
 * (see {@link #newInstance(String, ClassLoader, Map)} for the parameter semantics.)
 * or <tt>public static JAXBContext createContet(Class[],Map)</tt> method
 * (see {@link #newInstance(Class[], Map)} for the parameter semantics) are invoked
 * to create a {@link JAXBContext}.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see Marshaller
 * @see Unmarshaller
 * @see S 7.4.1 "Named Packages" in Java Language Specification</a>
 * @since JAXB1.0
 */
public abstract class JAXBContext {

    /**
     * The name of the property that contains the name of the class capable
     * of creating new <tt>JAXBContext</tt> objects.
     */
    public static final String JAXB_CONTEXT_FACTORY =
        "javax.xml.bind.context.factory";


    protected JAXBContext() {
    }


    /**
     * <p>
     * Obtain a new instance of a <tt>JAXBContext</tt> class.
     *
     * <p>
     * This is a convenience method to invoke the
     * {@link #newInstance(String,ClassLoader)} method with
     * the context class loader of the current thread.
     *
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>JAXBContext</tt> such as
     * <ol>
     *   <li>failure to locate either ObjectFactory.class or jaxb.index in the packages</li>
     *   <li>an ambiguity among global elements contained in the contextPath</li>
     *   <li>failure to locate a value for the context factory provider property</li>
```

```
     *   <li>mixing schema derived packages from different providers on the same contextPath</li>
     * </ol>
     */
    public static JAXBContext newInstance( String contextPath )
        throws JAXBException {

        //return newInstance( contextPath, JAXBContext.class.getClassLoader() );
        return newInstance( contextPath, getContextClassLoader());
    }

    /**
     * <p>
     * Obtain a new instance of a <tt>JAXBContext</tt> class.
     *
     * <p>
     * The client application must supply a context path which is a list of
     * colon (':', \u005Cu003A) separated java package names that contain
     * schema-derived classes and/or fully qualified JAXB-annotated classes.
     * Schema-derived
     * code is registered with the JAXBContext by the
     * ObjectFactory.class generated per package.
     * Alternatively than being listed in the context path, programmer
     * annotated JAXB mapped classes can be listed in a
     * <tt>jaxb.index</tt> resource file, format described below.
     * Note that a java package can contain both schema-derived classes and
     * user annotated JAXB classes. Additionally, the java package may
     * contain JAXB package annotations  that must be processed. (see JLS,
     * Section 7.4.1 "Named Packages").
     * </p>
     *
     * <p>
     * Every package listed on the contextPath must meet <b>one or both</b> of the
     * following conditions otherwise a <tt>JAXBException</tt> will be thrown:
     * </p>
     * <ol>
     *   <li>it must contain ObjectFactory.class</li>
     *   <li>it must contain jaxb.index</li>
     * </ol>
     *
     * <p>
     * <b>Format for jaxb.index</b>
     * <p>
     * The file contains a newline-separated list of class names.
     * Space and tab characters, as well as blank
     * lines, are ignored. The comment character
     * is '#' (0x23); on each line all characters following the first comment
     * character are ignored. The file must be encoded in UTF-8. Classes that
     * are reachable, as defined in {@link #newInstance(Class...)}, from the
     * listed classes are also registered with JAXBContext.
     * <p>
     * Constraints on class name occuring in a <tt>jaxb.index</tt> file are:
     * <ul>
     *   <li>Must not end with ".class".</li>
```

```
     *   <li>Class names are resolved relative to package containing
     *       <tt>jaxb.index</tt> file. Only classes occuring directly in package
     *       containing <tt>jaxb.index</tt> file are allowed.</li>
     *   <li>Fully qualified class names are not allowed.
     *       A qualified class name,relative to current package,
     *       is only allowed to specify a nested or inner class.</li>
     * </ul>
     *
     * <p>
     * To maintain compatibility with JAXB 1.0 schema to java
     * interface/implementation binding, enabled by schema customization
     * <tt>&lt;jaxb:globalBindings valueClass="false"></tt>,
     * the JAXB provider will ensure that each package on the context path
     * has a <tt>jaxb.properties</tt> file which contains a value for the
     * <tt>javax.xml.bind.context.factory</tt> property and that all values
     * resolve to the same provider.  This requirement does not apply to
     * JAXB annotated classes.
     *
     * <p>
     * If there are any global XML element name collisions across the various
     * packages listed on the <tt>contextPath</tt>, a <tt>JAXBException</tt>
     * will be thrown.
     *
     * <p>
     * Mixing generated interface/impl bindings from multiple JAXB Providers
     * in the same context path may result in a <tt>JAXBException</tt>
     * being thrown.
     *
     * <p>
     * The steps involved in discovering the JAXB implementation is discussed in the class javadoc.
     *
     * @param contextPath list of java package names that contain schema
     *                    derived class and/or java to schema (JAXB-annotated)
     *                    mapped classes
     * @param classLoader
     *      This class loader will be used to locate the implementation
     *      classes.
     *
     * @return a new instance of a <tt>JAXBContext</tt>
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>JAXBContext</tt> such as
     * <ol>
     *   <li>failure to locate either ObjectFactory.class or jaxb.index in the packages</li>
     *   <li>an ambiguity among global elements contained in the contextPath</li>
     *   <li>failure to locate a value for the context factory provider property</li>
     *   <li>mixing schema derived packages from different providers on the same contextPath</li>
     * </ol>
     */
    public static JAXBContext newInstance( String contextPath, ClassLoader classLoader ) throws
        JAXBException {

        return newInstance(contextPath,classLoader,Collections.<String,Object>emptyMap());
    }
```

```

    /**
     * <p>
     * Obtain a new instance of a <tt>JAXBContext</tt> class.
     *
     * <p>
     * This is mostly the same as {@link JAXBContext#newInstance(String, ClassLoader)},
     * but this version allows you to pass in provider-specific properties to configure
     * the instantiation of {@link JAXBContext}.
     *
     * <p>
     * The interpretation of properties is up to implementations. Implementations should
     * throw <tt>JAXBException</tt> if it finds properties that it doesn't understand.
     *
     * @param contextPath list of java package names that contain schema derived classes
     * @param classLoader
     *      This class loader will be used to locate the implementation classes.
     * @param properties
     *      provider-specific properties. Can be null, which means the same thing as passing
     *      in an empty map.
     *
     * @return a new instance of a <tt>JAXBContext</tt>
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>JAXBContext</tt> such as
     * <ol>
     *   <li>failure to locate either ObjectFactory.class or jaxb.index in the packages</li>
     *   <li>an ambiguity among global elements contained in the contextPath</li>
     *   <li>failure to locate a value for the context factory provider property</li>
     *   <li>mixing schema derived packages from different providers on the same contextPath</li>
     * </ol>
     * @since JAXB2.0
     */
    public static JAXBContext newInstance( String contextPath, ClassLoader classLoader, Map<String
        ,?>  properties  )
        throws JAXBException {

        return ContextFinder.find(
                        /* The default property name according to the JAXB spec */
                        JAXB_CONTEXT_FACTORY,

                        /* the context path supplied by the client app */
                        contextPath,

                        /* class loader to be used */
                        classLoader,
                        properties );
    }

// TODO: resurrect this once we introduce external annotations
//    /**
//     * <p>
//     * Obtain a new instance of a <tt>JAXBContext</tt> class.
//     *
```

```
//     * <p>
//     * The client application must supply a list of classes that the new
//     * context object needs to recognize.
//     *
//     * Not only the new context will recognize all the classes specified,
//     * but it will also recognize any classes that are directly/indirectly
//     * referenced statically from the specified classes.
//     *
//     * For example, in the following Java code, if you do
//     * <tt>newInstance(Foo.class)</tt>, the newly created {@link JAXBContext}
//     * will recognize both <tt>Foo</tt> and <tt>Bar</tt>, but not <tt>Zot</tt>:
//     * <pre>
//     * class Foo {
//     *      Bar b;
//     * }
//     * class Bar { int x; }
//     * class Zot extends Bar { int y; }
//     * </pre>
//     *
//     * Therefore, a typical client application only needs to specify the
//     * top-level classes, but it needs to be careful.
//     *
//     * TODO: if we are to define other mechanisms, refer to them.
//     *
//     * @param externalBindings
//     *      list of external binding files. Can be null or empty if none is used.
//     *      when specified, those files determine how the classes are bound.
//     *
//     * @param classesToBeBound
//     *      list of java classes to be recognized by the new {@link JAXBContext}.
//     *      Can be empty, in which case a {@link JAXBContext} that only knows about
//     *      spec-defined classes will be returned.
//     *
//     * @return
//     *      A new instance of a <tt>JAXBContext</tt>. Always non-null valid object.
//     *
//     * @throws JAXBException
//     *      if an error was encountered while creating the
//     *      <tt>JAXBContext</tt>, such as (but not limited to):
//     * <ol>
//     *  <li>No JAXB implementation was discovered
//     *  <li>Classes use JAXB annotations incorrectly
//     *  <li>Classes have colliding annotations (i.e., two classes with the same type name)
//     *  <li>Specified external bindings are incorrect
//     *  <li>The JAXB implementation was unable to locate
//     *      provider-specific out-of-band information (such as additional
//     *      files generated at the development time.)
//     * </ol>
//     *
//     * @throws IllegalArgumentException
//     *      if the parameter contains {@code null} (i.e., {@code newInstance(null);})
//     *
//     * @since JAXB2.0
```

```
//     */
//    public static JAXBContext newInstance( Source[] externalBindings, Class... classesToBeBound )
//        throws JAXBException {
//
//        // empty class list is not an error, because the context will still include
//        // spec-specified classes like String and Integer.
//        // if(classesToBeBound.length==0)
//        //    throw new IllegalArgumentException();
//
//        // but it is an error to have nulls in it.
//        for( int i=classesToBeBound.length-1; i>=0; i-- )
//            if(classesToBeBound[i]==null)
//                throw new IllegalArgumentException();
//
//        return ContextFinder.find(externalBindings,classesToBeBound);
//    }

    /**
     * <p>
     * Obtain a new instance of a <tt>JAXBContext</tt> class.
     *
     * <p>
     * The client application must supply a list of classes that the new
     * context object needs to recognize.
     *
     * Not only the new context will recognize all the classes specified,
     * but it will also recognize any classes that are directly/indirectly
     * referenced statically from the specified classes. Subclasses of
     * referenced classes nor <tt>&#64;XmlTransient</tt> referenced classes
     * are not registered with JAXBContext.
     *
     * For example, in the following Java code, if you do
     * <tt>newInstance(Foo.class)</tt>, the newly created {@link JAXBContext}
     * will recognize both <tt>Foo</tt> and <tt>Bar</tt>, but not <tt>Zot</tt> or <tt>FooBar</tt>:
     * <pre>
     * class Foo {
     *      &#64;XmlTransient FooBar c;
     *      Bar b;
     * }
     * class Bar { int x; }
     * class Zot extends Bar { int y; }
     * class FooBar { }
     * </pre>
     *
     * Therefore, a typical client application only needs to specify the
     * top-level classes, but it needs to be careful.
     *
     * <p>
     * Note that for each java package registered with JAXBContext,
     * when the optional package annotations exist, they must be processed.
     * (see JLS, Section 7.4.1 "Named Packages").
     *
     * <p>
```

```
     * The steps involved in discovering the JAXB implementation is discussed in the class javadoc.
     *
     * @param classesToBeBound
     *      list of java classes to be recognized by the new {@link JAXBContext}.
     *      Can be empty, in which case a {@link JAXBContext} that only knows about
     *      spec-defined classes will be returned.
     *
     * @return
     *      A new instance of a <tt>JAXBContext</tt>. Always non-null valid object.
     *
     * @throws JAXBException
     *      if an error was encountered while creating the
     *      <tt>JAXBContext</tt>, such as (but not limited to):
     * <ol>
     *  <li>No JAXB implementation was discovered
     *  <li>Classes use JAXB annotations incorrectly
     *  <li>Classes have colliding annotations (i.e., two classes with the same type name)
     *  <li>The JAXB implementation was unable to locate
     *      provider-specific out-of-band information (such as additional
     *      files generated at the development time.)
     * </ol>
     *
     * @throws IllegalArgumentException
     *      if the parameter contains {@code null} (i.e., {@code newInstance(null);})
     *
     * @since JAXB2.0
     */
    public static JAXBContext newInstance( Class... classesToBeBound )
        throws JAXBException {

        return newInstance(classesToBeBound,Collections.<String,Object>emptyMap());
    }

    /**
     * <p>
     * Obtain a new instance of a <tt>JAXBContext</tt> class.
     *
     * <p>
     * An overloading of {@link JAXBContext#newInstance(Class...)}
     * to configure 'properties' for this instantiation of {@link JAXBContext}.
     *
     * <p>
     * The interpretation of properties is up to implementations. Implementations should
     * throw <tt>JAXBException</tt> if it finds properties that it doesn't understand.
     *
     * @param classesToBeBound
     *      list of java classes to be recognized by the new {@link JAXBContext}.
     *      Can be empty, in which case a {@link JAXBContext} that only knows about
     *      spec-defined classes will be returned.
     * @param properties
     *      provider-specific properties. Can be null, which means the same thing as passing
     *      in an empty map.
     *
```

```
     * @return
     *      A new instance of a <tt>JAXBContext</tt>. Always non-null valid object.
     *
     * @throws JAXBException
     *      if an error was encountered while creating the
     *      <tt>JAXBContext</tt>, such as (but not limited to):
     * <ol>
     *  <li>No JAXB implementation was discovered
     *  <li>Classes use JAXB annotations incorrectly
     *  <li>Classes have colliding annotations (i.e., two classes with the same type name)
     *  <li>The JAXB implementation was unable to locate
     *      provider-specific out-of-band information (such as additional
     *      files generated at the development time.)
     * </ol>
     *
     * @throws IllegalArgumentException
     *      if the parameter contains {@code null} (i.e., {@code newInstance(null,someMap);})
     *
     * @since JAXB2.0
     */
    public static JAXBContext newInstance( Class[] classesToBeBound, Map<String,?> properties )
        throws JAXBException {

        if (classesToBeBound == null) {
                throw new IllegalArgumentException();
        }

        // but it is an error to have nulls in it.
        for (int i = classesToBeBound.length - 1; i >= 0; i--) {
            if (classesToBeBound[i] == null) {
                throw new IllegalArgumentException();
            }
        }

        return ContextFinder.find(classesToBeBound,properties);
    }

    /**
     * Create an <tt>Unmarshaller</tt> object that can be used to convert XML
     * data into a java content tree.
     *
     * @return an <tt>Unmarshaller</tt> object
     *
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>Unmarshaller</tt> object
     */
    public abstract Unmarshaller createUnmarshaller() throws JAXBException;


    /**
     * Create a <tt>Marshaller</tt> object that can be used to convert a
     * java content tree into XML data.
     *
```

```
     * @return a <tt>Marshaller</tt> object
     *
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>Marshaller</tt> object
     */
    public abstract Marshaller createMarshaller() throws JAXBException;


    /**
     * {@link Validator} has been made optional and deprecated in JAXB 2.0.  Please
     * refer to the javadoc for {@link Validator} for more detail.
     * <p>
     * Create a <tt>Validator</tt> object that can be used to validate a
     * java content tree against its source schema.
     *
     * @return a <tt>Validator</tt> object
     *
     * @throws JAXBException if an error was encountered while creating the
     *                       <tt>Validator</tt> object
     * @deprecated since JAXB2.0
     */
    public abstract Validator createValidator() throws JAXBException;

    /**
     * Creates a <tt>Binder</tt> object that can be used for
     * associative/in-place unmarshalling/marshalling.
     *
     * @param domType select the DOM API to use by passing in its DOM Node class.
     *
     * @return always a new valid <tt>Binder</tt> object.
     *
     * @throws UnsupportedOperationException
     *      if DOM API corresponding to <tt>domType</tt> is not supported by
     *      the implementation.
     *
     * @since JAXB2.0
     */
    public <T> Binder<T> createBinder(Class<T> domType) {
        // to make JAXB 1.0 implementations work, this method must not be
        // abstract
        throw new UnsupportedOperationException();
    }

    /**
     * Creates a <tt>Binder</tt> for W3C DOM.
     *
     * @return always a new valid <tt>Binder</tt> object.
     *
     * @since JAXB2.0
     */
    public Binder<Node> createBinder() {
        return createBinder(Node.class);
    }
```

```

    /**
     * Creates a <tt>JAXBIntrospector</tt> object that can be used to
     * introspect JAXB objects.
     *
     * @return
     *      always return a non-null valid <tt>JAXBIntrospector</tt> object.
     *
     * @throws UnsupportedOperationException
     *      Calling this method on JAXB 1.0 implementations will throw
     *      an UnsupportedOperationException.
     *
     * @since JAXB2.0
     */
    public JAXBIntrospector createJAXBIntrospector() {
        // to make JAXB 1.0 implementations work, this method must not be
        // abstract
        throw new UnsupportedOperationException();
    }

    /**
     * Generates the schema documents for this context.
     *
     * @param outputResolver
     *      this object controls the output to which schemas
     *      will be sent.
     *
     * @throws IOException
     *      if {@link SchemaOutputResolver} throws an {@link IOException}.
     *
     * @throws UnsupportedOperationException
     *      Calling this method on JAXB 1.0 implementations will throw
     *      an UnsupportedOperationException.
     *
     * @since JAXB 2.0
     */
    public void generateSchema(SchemaOutputResolver outputResolver) throws IOException  {
        // to make JAXB 1.0 implementations work, this method must not be
        // abstract
        throw new UnsupportedOperationException();
    }

    private static ClassLoader getContextClassLoader() {
        if (System.getSecurityManager() == null) {
            return Thread.currentThread().getContextClassLoader();
        } else {
            return (ClassLoader) java.security.AccessController.doPrivileged(
                    new java.security.PrivilegedAction() {
                        public java.lang.Object run() {
                            return Thread.currentThread().getContextClassLoader();
                        }
                    });
        }
```

```
    }

}
```

2.11 JAXBElement.java

Secuencia 13: javax/xml/bind/JAXBElement.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import javax.xml.namespace.QName;
import java.io.Serializable;

/**
 * <p>JAXB representation of an Xml Element.</p>
 *
 * <p>This class represents information about an Xml Element from both the element
 * declaration within a schema and the element instance value within an xml document
 * with the following properties
 * <ul>
 *   <li>element's xml tag <b><tt>name</tt></b></li>
 *   <li><b><tt>value</tt></b> represents the element instance's atttribute(s) and content model</li>
 *   <li>element declaration's <b><tt>declaredType</tt></b> (<tt>xs:element @type</tt> attribute)</li>
 *   <li><b><tt>scope</tt></b> of element declaration</li>
 *   <li>boolean <b><tt>nil</tt></b> property. (element instance's <tt><b>xsi:nil</b></tt> attribute)</li>
 * </ul>
```

```
 *
 * <p>The <tt>declaredType</tt> and <tt>scope</tt> property are the
 * JAXB class binding for the xml type definition.
 * </p>
 *
 * <p><b><tt>Scope</tt></b> is either {@link GlobalScope} or the Java class representing the
 * complex type definition containing the schema element declaration.
 * </p>
 *
 * <p>There is a property constraint that if <b><tt>value</tt></b> is <tt>null</tt>,
 * then <tt>nil</tt> must be <tt>true</tt>. The converse is not true to enable
 * representing a nil element with attribute(s). If <tt>nil</tt> is true, it is possible
 * that <tt>value</tt> is non-null so it can hold the value of the attributes
 * associated with a nil element.
 * </p>
 *
 * @author Kohsuke Kawaguchi, Joe Fialli
 * @since JAXB 2.0
 */

public class JAXBElement<T> implements Serializable {

    /** xml element tag name */
    final protected QName name;

    /** Java datatype binding for xml element declaration's type. */
    final protected Class<T> declaredType;

    /** Scope of xml element declaration representing this xml element instance.
     *  Can be one of the following values:
     *  - {@link GlobalScope} for global xml element declaration.
     *  - local element declaration has a scope set to the Java class
     *     representation of complex type defintion containing
     *     xml element declaration.
     */
    final protected Class scope;

    /** xml element value.
        Represents content model and attributes of an xml element instance. */
    protected T value;

    /** true iff the xml element instance has xsi:nil="true". */
    protected boolean nil = false;

    /**
     * Designates global scope for an xml element.
     */
    public static final class GlobalScope {}

    /**
     * <p>Construct an xml element instance.</p>
     *
     * @param name          Java binding of xml element tag name
```

```
     * @param declaredType  Java binding of xml element declaration's type
     * @param scope
     *      Java binding of scope of xml element declaration.
     *      Passing null is the same as passing <tt>GlobalScope.class</tt>
     * @param value
     *      Java instance representing xml element's value.
     * @see #getScope()
     * @see #isTypeSubstituted()
     */
    public JAXBElement(QName name,
                       Class<T> declaredType,
                       Class scope,
                       T value) {
        if(declaredType==null || name==null)
            throw new IllegalArgumentException();
        this.declaredType = declaredType;
        if(scope==null)     scope = GlobalScope.class;
        this.scope = scope;
        this.name = name;
        setValue(value);
    }

    /**
     * Construct an xml element instance.
     *
     * This is just a convenience method for <tt>new JAXBElement(name,declaredType,GlobalScope.class,value)</tt>
     */
    public JAXBElement(QName name, Class<T> declaredType, T value ) {
        this(name,declaredType,GlobalScope.class,value);
    }

    /**
     * Returns the Java binding of the xml element declaration's type attribute.
     */
    public Class<T> getDeclaredType() {
        return declaredType;
    }

    /**
     * Returns the xml element tag name.
     */
    public QName getName() {
        return name;
    }

    /**
     * <p>Set the content model and attributes of this xml element.</p>
     *
     * <p>When this property is set to <tt>null</tt>, <tt>isNil()</tt> must by <tt>true</tt>.
     *    Details of constraint are described at {@link #isNil()}.</p>
     *
     * @see #isTypeSubstituted()
```

```
     */
    public void setValue(T t) {
        this.value = t;
    }

    /**
     * <p>Return the content model and attribute values for this element.</p>
     *
     * <p>See {@link #isNil()} for a description of a property constraint when
     * this value is <tt>null</tt></p>
     */
    public T getValue() {
        return value;
    }

    /**
     * Returns scope of xml element declaration.
     *
     * @see #isGlobalScope()
     * @return <tt>GlobalScope.class</tt> if this element is of global scope.
     */
    public Class getScope() {
        return scope;
    }

    /**
     * <p>Returns <tt>true</tt> iff this element instance content model
     * is nil.</p>
     *
     * <p>This property always returns <tt>true</tt> when {@link #getValue()} is null.
     * Note that the converse is not true, when this property is <tt>true</tt>,
     * {@link #getValue()} can contain a non-null value for attribute(s). It is
     * valid for a nil xml element to have attribute(s).</p>
     */
    public boolean isNil() {
        return (value == null) || nil;
    }

    /**
     * <p>Set whether this element has nil content.</p>
     *
     * @see #isNil()
     */
    public void setNil(boolean value) {
        this.nil = value;
    }

    /* Convenience methods
     * (Not necessary but they do unambiguously conceptualize
     *  the rationale behind this class' fields.)
     */

    /**
```

```
     * Returns true iff this xml element declaration is global.
     */
    public boolean isGlobalScope() {
        return this.scope == GlobalScope.class;
    }

    /**
     * Returns true iff this xml element instance's value has a different
     * type than xml element declaration's declared type.
     */
    public boolean isTypeSubstituted() {
        if(value==null)     return false;
        return value.getClass() != declaredType;
    }

    private static final long serialVersionUID = 1L;
}
```

2.12 JAXBException.java

Secuencia 14: javax/xml/bind/JAXBException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import java.io.PrintWriter;

/**
 * This is the root exception class for all JAXB exceptions.
 *
```

```
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 * @see Marshaller
 * @see Unmarshaller
 * @since JAXB1.0
 */
public class JAXBException extends Exception {

    /**
     * Vendor specific error code
     *
     */
    private String errorCode;

    /**
     * Exception reference
     *
     */
    private volatile Throwable linkedException;

    static final long serialVersionUID = -5621384651494307979L;

    /**
     * Construct a JAXBException with the specified detail message.  The
     * errorCode and linkedException will default to null.
     *
     * @param message a description of the exception
     */
    public JAXBException(String message) {
        this( message, null, null );
    }

    /**
     * Construct a JAXBException with the specified detail message and vendor
     * specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public JAXBException(String message, String errorCode) {
        this( message, errorCode, null );
    }

    /**
     * Construct a JAXBException with a linkedException.  The detail message and
     * vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public JAXBException(Throwable exception) {
        this( null, null, exception );
    }

```

```java
    /**
     * Construct a JAXBException with the specified detail message and
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public JAXBException(String message, Throwable exception) {
        this( message, null, exception );
    }

    /**
     * Construct a JAXBException with the specified detail message, vendor
     * specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public JAXBException(String message, String errorCode, Throwable exception) {
        super( message );
        this.errorCode = errorCode;
        this.linkedException = exception;
    }

    /**
     * Get the vendor specific error code
     *
     * @return a string specifying the vendor specific error code
     */
    public String getErrorCode() {
        return this.errorCode;
    }

    /**
     * Get the linked exception
     *
     * @return the linked Exception, null if none exists
     */
    public Throwable getLinkedException() {
        return linkedException;
    }

    /**
     * Add a linked Exception.
     *
     * @param exception the linked Exception (A null value is permitted and
     *                  indicates that the linked exception does not exist or
     *                  is unknown).
     */
    public void setLinkedException( Throwable exception ) {
        this.linkedException = exception;
    }
```

```

    /**
     * Returns a short description of this JAXBException.
     *
     */
    public String toString() {
        return linkedException == null ?
            super.toString() :
            super.toString() + "\n - with linked exception:\n[" +
                                linkedException.toString()+ "]";
    }

    /**
     * Prints this JAXBException and its stack trace (including the stack trace
     * of the linkedException if it is non-null) to the PrintStream.
     *
     * @param s PrintStream to use for output
     */
    public void printStackTrace( java.io.PrintStream s ) {
        super.printStackTrace(s);
    }

    /**
     * Prints this JAXBException and its stack trace (including the stack trace
     * of the linkedException if it is non-null) to <tt>System.err</tt>.
     *
     */
    public void printStackTrace() {
        super.printStackTrace();
    }

    /**
     * Prints this JAXBException and its stack trace (including the stack trace
     * of the linkedException if it is non-null) to the PrintWriter.
     *
     * @param s PrintWriter to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
    }

    @Override
    public Throwable getCause() {
        return linkedException;
    }
}
```

2.13 JAXBIntrospector.java

Secuencia 15: javax/xml/bind/JAXBIntrospector.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;
import  javax.xml.namespace.QName;

/**
 * Provide access to JAXB xml binding data for a JAXB object.
 *
 * <p>
 * Intially, the intent of this class is to just conceptualize how
 * a JAXB application developer can access xml binding information,
 * independent if binding model is java to schema or schema to java.
 * Since accessing the XML element name related to a JAXB element is
 * a highly requested feature, demonstrate access to this
 * binding information.
 *
 * The factory method to get a <code>JAXBIntrospector</code> instance is
 * {@link JAXBContext#createJAXBIntrospector()}.
 *
 * @see JAXBContext#createJAXBIntrospector()
 * @since JAXB2.0
 */
public abstract class JAXBIntrospector {

    /**
     * <p>Return true if <code>object</code> represents a JAXB element.</p>
     * <p>Parameter <code>object</code> is a JAXB element for following cases:
     * <ol>
     *   <li>It is an instance of <code>javax.xml.bind.JAXBElement</code>.</li>
     *   <li>The class of <code>object</code> is annotated with
     *       <code>&#64XmlRootElement</code>.
     *   </li>
     * </ol>
```

```
     *
     * @see #getElementName(Object)
     */
    public abstract boolean isElement(Object object);

    /**
     * <p>Get xml element qname for <code>jaxbElement</code>.</p>
     *
     * @param jaxbElement is an object that {@link #isElement(Object)} returned true.
     *
     * @return xml element qname associated with jaxbElement;
     *         null if <code>jaxbElement</code> is not a JAXB Element.
     */
    public abstract QName getElementName(Object jaxbElement);

    /**
     * <p>Get the element value of a JAXB element.</p>
     *
     * <p>Convenience method to abstract whether working with either
     *    a javax.xml.bind.JAXBElement instance or an instance of
     *    <tt>&#64XmlRootElement</tt> annotated Java class.</p>
     *
     * @param jaxbElement  object that #isElement(Object) returns true.
     *
     * @return The element value of the <code>jaxbElement</code>.
     */
    public static Object getValue(Object jaxbElement) {
        if (jaxbElement instanceof JAXBElement) {
            return ((JAXBElement)jaxbElement).getValue();
        } else {
            // assume that class of this instance is
            // annotated with @XmlRootElement.
            return jaxbElement;
        }
    }
}
```

2.14 JAXBPermission.java

Secuencia 16: javax/xml/bind/JAXBPermission.java

```
/*
 * Copyright (c) 2007, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import java.security.BasicPermission;

/**
 * This class is for JAXB permissions. A {@code JAXBPermission}
 * contains a name (also referred to as a "target name") but
 * no actions list; you either have the named permission
 * or you don't.
 *
 * <P>
 * The target name is the name of the JAXB permission (see below).
 *
 * <P>
 * The following table lists all the possible {@code JAXBPermission} target names,
 * and for each provides a description of what the permission allows
 * and a discussion of the risks of granting code the permission.
 * <P>
 *
 * <table border=1 cellpadding=5 summary="Permission target name, what the permission allows, and
     associated risks">
 * <tr>
 * <th>Permission Target Name</th>
 * <th>What the Permission Allows</th>
 * <th>Risks of Allowing this Permission</th>
 * </tr>
 *
 * <tr>
 *   <td>setDatatypeConverter</td>
 *   <td>
 *     Allows the code to set VM-wide {@link DatatypeConverterInterface}
 *     via {@link DatatypeConverter#setDatatypeConverter(DatatypeConverterInterface) the
     setDatatypeConverter method}
 *     that all the methods on {@link DatatypeConverter} uses.
 *   </td>
 *   <td>
 *     Malicious code can set {@link DatatypeConverterInterface}, which has
 *     VM-wide singleton semantics,  before a genuine JAXB implementation sets one.
 *     This allows malicious code to gain access to objects that it may otherwise
 *     not have access to, such as {@link java.awt.Frame#getFrames()} that belongs to
 *     another application running in the same JVM.
```

```
 *   </td>
 * </tr>
 * </table>
 *
 * @see java.security.BasicPermission
 * @see java.security.Permission
 * @see java.security.Permissions
 * @see java.security.PermissionCollection
 * @see java.lang.SecurityManager
 *
 * @author Joe Fialli
 * @since JAXB 2.2
 */

/* code was borrowed originally from java.lang.RuntimePermission. */
public final class JAXBPermission extends BasicPermission {
    /**
     * Creates a new JAXBPermission with the specified name.
     *
     * @param name
     * The name of the JAXBPermission. As of 2.2 only "setDatatypeConverter"
     * is defined.
     */
    public JAXBPermission(String name) {
        super(name);
    }

    private static final long serialVersionUID = 1L;
}
```

2.15 MarshalException.java

Secuencia 17: javax/xml/bind/MarshalException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This exception indicates that an error has occurred while performing
 * a marshal operation that the provider is unable to recover from.
 *
 * <p>
 * The <tt>ValidationEventHandler</tt> can cause this exception to be thrown
 * during the marshal operations.  See
 * {@link ValidationEventHandler#handleEvent(ValidationEvent)
 * ValidationEventHandler.handleEvent(ValidationEvent)}.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see JAXBException
 * @see Marshaller
 * @since JAXB1.0
 */
public class MarshalException extends JAXBException {

    /**
     * Construct a MarshalException with the specified detail message.  The
     * errorCode and linkedException will default to null.
     *
     * @param message a description of the exception
     */
    public MarshalException( String message ) {
        this( message, null, null );
    }

    /**
     * Construct a MarshalException with the specified detail message and vendor
     * specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public MarshalException( String message, String errorCode ) {
        this( message, errorCode, null );
    }

    /**
     * Construct a MarshalException with a linkedException.  The detail message and
     * vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public MarshalException( Throwable exception ) {
        this( null, null, exception );
```

```
    }

    /**
     * Construct a MarshalException with the specified detail message and
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public MarshalException( String message, Throwable exception ) {
        this( message, null, exception );
    }

    /**
     * Construct a MarshalException with the specified detail message, vendor
     * specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public MarshalException( String message, String errorCode, Throwable exception ) {
        super( message, errorCode, exception );
    }

}
```

2.16 Marshaller.java

Secuencia 18: javax/xml/bind/Marshaller.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

package javax.xml.bind;

import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.attachment.AttachmentMarshaller;
import javax.xml.validation.Schema;
import java.io.File;

/**
 * <p>
 * The <tt>Marshaller</tt> class is responsible for governing the process
 * of serializing Java content trees back into XML data.  It provides the basic
 * marshalling methods:
 *
 * <p>
 * <i>Assume the following setup code for all following code fragments:</i>
 * <blockquote>
 *    <pre>
 *       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
 *       Unmarshaller u = jc.createUnmarshaller();
 *       Object element = u.unmarshal( new File( "foo.xml" ) );
 *       Marshaller m = jc.createMarshaller();
 *    </pre>
 * </blockquote>
 *
 * <p>
 * Marshalling to a File:
 * <blockquote>
 *    <pre>
 *       OutputStream os = new FileOutputStream( "nosferatu.xml" );
 *       m.marshal( element, os );
 *    </pre>
 * </blockquote>
 *
 * <p>
 * Marshalling to a SAX ContentHandler:
 * <blockquote>
 *    <pre>
 *       // assume MyContentHandler instanceof ContentHandler
 *       m.marshal( element, new MyContentHandler() );
 *    </pre>
 * </blockquote>
 *
 * <p>
 * Marshalling to a DOM Node:
 * <blockquote>
 *    <pre>
 *       DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
 *       dbf.setNamespaceAware(true);
 *       DocumentBuilder db = dbf.newDocumentBuilder();
 *       Document doc = db.newDocument();
```

```
*
*       m.marshal( element, doc );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a java.io.OutputStream:
* <blockquote>
*    <pre>
*       m.marshal( element, System.out );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a java.io.Writer:
* <blockquote>
*    <pre>
*       m.marshal( element, new PrintWriter( System.out ) );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a javax.xml.transform.SAXResult:
* <blockquote>
*    <pre>
*       // assume MyContentHandler instanceof ContentHandler
*       SAXResult result = new SAXResult( new MyContentHandler() );
*
*       m.marshal( element, result );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a javax.xml.transform.DOMResult:
* <blockquote>
*    <pre>
*       DOMResult result = new DOMResult();
*
*       m.marshal( element, result );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a javax.xml.transform.StreamResult:
* <blockquote>
*    <pre>
*       StreamResult result = new StreamResult( System.out );
*
*       m.marshal( element, result );
*    </pre>
* </blockquote>
*
* <p>
```

```
* Marshalling to a javax.xml.stream.XMLStreamWriter:
* <blockquote>
*    <pre>
*       XMLStreamWriter xmlStreamWriter =
*           XMLOutputFactory.newInstance().createXMLStreamWriter( ... );
*
*       m.marshal( element, xmlStreamWriter );
*    </pre>
* </blockquote>
*
* <p>
* Marshalling to a javax.xml.stream.XMLEventWriter:
* <blockquote>
*    <pre>
*       XMLEventWriter xmlEventWriter =
*           XMLOutputFactory.newInstance().createXMLEventWriter( ... );
*
*       m.marshal( element, xmlEventWriter );
*    </pre>
* </blockquote>
*
* <p>
* <a name="elementMarshalling"></a>
* <b>Marshalling content tree rooted by a JAXB element</b><br>
* <blockquote>
* The first parameter of the overloaded
* <tt>Marshaller.marshal(java.lang.Object, ...)</tt> methods must be a
* JAXB element as computed by
* {@link JAXBIntrospector#isElement(java.lang.Object)};
* otherwise, a <tt>Marshaller.marshal</tt> method must throw a
* {@link MarshalException}. There exist two mechanisms
* to enable marshalling an instance that is not a JAXB element.
* One method is to wrap the instance as a value of a {@link JAXBElement},
* and pass the wrapper element as the first parameter to
* a <tt>Marshaller.marshal</tt> method. For java to schema binding, it
* is also possible to simply annotate the instance's class with
* &#64;{@link XmlRootElement}.
* </blockquote>
*
* <p>
* <b>Encoding</b><br>
* <blockquote>
* By default, the Marshaller will use UTF-8 encoding when generating XML data
* to a <tt>java.io.OutputStream</tt>, or a <tt>java.io.Writer</tt>.  Use the
* {@link #setProperty(String,Object) setProperty} API to change the output
* encoding used during these marshal operations.  Client applications are
* expected to supply a valid character encoding name as defined in the
* <a href="http://www.w3.org/TR/2000/REC-xml-20001006#charencoding">W3C XML 1.0
* Recommendation</a> and supported by your Java Platform</a>.
* </blockquote>
*
* <p>
* <b>Validation and Well-Formedness</b><br>
```

```
* <blockquote>
* <p>
* Client applications are not required to validate the Java content tree prior
* to calling any of the marshal API's.  Furthermore, there is no requirement
* that the Java content tree be valid with respect to its original schema in
* order to marshal it back into XML data.  Different JAXB Providers will
* support marshalling invalid Java content trees at varying levels, however
* all JAXB Providers must be able to marshal a valid content tree back to
* XML data.  A JAXB Provider must throw a <tt>MarshalException</tt> when it
* is unable to complete the marshal operation due to invalid content.  Some
* JAXB Providers will fully allow marshalling invalid content, others will fail
* on the first validation error.
* <p>
* Even when schema validation is not explictly enabled for the marshal operation,
* it is possible that certain types of validation events will be detected
* during the operation.  Validation events will be reported to the registered
* event handler.  If the client application has not registered an event handler
* prior to invoking one of the marshal API's, then events will be delivered to
* a default event handler which will terminate the marshal operation after
* encountering the first error or fatal error. Note that for JAXB 2.0 and
* later versions, {@link javax.xml.bind.helpers.DefaultValidationEventHandler} is
* no longer used.
*
* </blockquote>
*
* <p>
* <a name="supportedProps"></a>
* <b>Supported Properties</b><br>
* <blockquote>
* <p>
* All JAXB Providers are required to support the following set of properties.
* Some providers may support additional properties.
* <dl>
*   <dt><tt>jaxb.encoding</tt> - value must be a java.lang.String</dt>
*   <dd>The output encoding to use when marshalling the XML data.  The
*               Marshaller will use "UTF-8" by default if this property is not
*       specified.</dd>
*   <dt><tt>jaxb.formatted.output</tt> - value must be a java.lang.Boolean</dt>
*   <dd>This property controls whether or not the Marshaller will format
*       the resulting XML data with line breaks and indentation.  A
*       true value for this property indicates human readable indented
*       xml data, while a false value indicates unformatted xml data.
*       The Marshaller will default to false (unformatted) if this
*       property is not specified.</dd>
*   <dt><tt>jaxb.schemaLocation</tt> - value must be a java.lang.String</dt>
*   <dd>This property allows the client application to specify an
*       xsi:schemaLocation attribute in the generated XML data.  The format of
*       the schemaLocation attribute value is discussed in an easy to
*       understand, non-normative form in
*       <a href="http://www.w3.org/TR/xmlschema-0/#schemaLocation">Section 5.6
*       of the W3C XML Schema Part 0: Primer</a> and specified in
*       <a href="http://www.w3.org/TR/xmlschema-1/#Instance_Document_Constructions">
*       Section 2.6 of the W3C XML Schema Part 1: Structures</a>.</dd>
```

```
*   <dt><tt>jaxb.noNamespaceSchemaLocation</tt> - value must be a java.lang.String</dt>
*   <dd>This property allows the client application to specify an
*       xsi:noNamespaceSchemaLocation attribute in the generated XML
*       data.  The format of the schemaLocation attribute value is discussed in
*       an easy to understand, non-normative form in
*       <a href="http://www.w3.org/TR/xmlschema-0/#schemaLocation">Section 5.6
*       of the W3C XML Schema Part 0: Primer</a> and specified in
*       <a href="http://www.w3.org/TR/xmlschema-1/#Instance_Document_Constructions">
*       Section 2.6 of the W3C XML Schema Part 1: Structures</a>.</dd>
*   <dt><tt>jaxb.fragment</tt> - value must be a java.lang.Boolean</dt>
*   <dd>This property determines whether or not document level events will be
*       generated by the Marshaller.  If the property is not specified, the
*       default is <tt>false</tt>. This property has different implications depending
*       on which marshal api you are using - when this property is set to true:<br>
*       <ul>
*         <li>{@link #marshal(Object,org.xml.sax.ContentHandler) marshal(Object,ContentHandler)} - the Marshaller won't
*             invoke {@link org.xml.sax.ContentHandler#startDocument()} and
*             {@link org.xml.sax.ContentHandler#endDocument()}.</li>
*         <li>{@link #marshal(Object,org.w3c.dom.Node) marshal(Object,Node)} - the property has no effect on this
*             API.</li>
*         <li>{@link #marshal(Object,java.io.OutputStream) marshal(Object,OutputStream)} - the Marshaller won't
*             generate an xml declaration.</li>
*         <li>{@link #marshal(Object,java.io.Writer) marshal(Object,Writer)} - the Marshaller won't
*             generate an xml declaration.</li>
*         <li>{@link #marshal(Object,javax.xml.transform.Result) marshal(Object,Result)} - depends on the kind of
*             Result object, see semantics for Node, ContentHandler, and Stream APIs</li>
*         <li>{@link #marshal(Object,javax.xml.stream.XMLEventWriter) marshal(Object,XMLEventWriter)} - the
*             Marshaller will not generate {@link javax.xml.stream.events.XMLEvent#START_DOCUMENT} and
*             {@link javax.xml.stream.events.XMLEvent#END_DOCUMENT} events.</li>
*         <li>{@link #marshal(Object,javax.xml.stream.XMLStreamWriter) marshal(Object,XMLStreamWriter)} - the
*             Marshaller will not generate {@link javax.xml.stream.events.XMLEvent#START_DOCUMENT} and
*             {@link javax.xml.stream.events.XMLEvent#END_DOCUMENT} events.</li>
*       </ul>
*   </dd>
* </dl>
* </blockquote>
*
* <p>
* <a name="marshalEventCallback"></a>
* <b>Marshal Event Callbacks</b><br>
* <blockquote>
* "The {@link Marshaller} provides two styles of callback mechanisms
* that allow application specific processing during key points in the
* unmarshalling process.  In 'class defined' event callbacks, application
```

```
 * specific code placed in JAXB mapped classes is triggered during
 * marshalling.  'External listeners' allow for centralized processing
 * of marshal events in one callback method rather than by type event callbacks.
 *
 * <p>
 * Class defined event callback methods allow any JAXB mapped class to specify
 * its own specific callback methods by defining methods with the following method signatures:
 * <blockquote>
 * <pre>
 *   // Invoked by Marshaller after it has created an instance of this object.
 *   boolean beforeMarshal(Marshaller);
 *
 *   // Invoked by Marshaller after it has marshalled all properties of this object.
 *   void afterMarshal(Marshaller);
 * </pre>
 * </blockquote>
 * The class defined event callback methods should be used when the callback method requires
 * access to non-public methods and/or fields of the class.
 * <p>
 * The external listener callback mechanism enables the registration of a {@link Listener}
 * instance with a {@link Marshaller#setListener(Listener)}. The external listener receives all callback events,
 * allowing for more centralized processing than per class defined callback methods.
 * <p>
 * The 'class defined' and external listener event callback methods are independent of each other,
 * both can be called for one event. The invocation ordering when both listener callback methods exist is
 * defined in {@link Listener#beforeMarshal(Object)} and {@link Listener#afterMarshal(Object)}.
 * <p>
 * An event callback method throwing an exception terminates the current marshal process.
 * </blockquote>
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 * @see Validator
 * @see Unmarshaller
 * @since JAXB1.0
 */
public interface Marshaller {

    /**
     * The name of the property used to specify the output encoding in
     * the marshalled XML data.
     */
    public static final String JAXB_ENCODING =
        "jaxb.encoding";

    /**
     * The name of the property used to specify whether or not the marshalled
     * XML data is formatted with linefeeds and indentation.
     */
    public static final String JAXB_FORMATTED_OUTPUT =
```

```
        "jaxb.formatted.output";

    /**
     * The name of the property used to specify the xsi:schemaLocation
     * attribute value to place in the marshalled XML output.
     */
    public static final String JAXB_SCHEMA_LOCATION =
        "jaxb.schemaLocation";

    /**
     * The name of the property used to specify the
     * xsi:noNamespaceSchemaLocation attribute value to place in the marshalled
     * XML output.
     */
    public static final String JAXB_NO_NAMESPACE_SCHEMA_LOCATION =
        "jaxb.noNamespaceSchemaLocation";

    /**
     * The name of the property used to specify whether or not the marshaller
     * will generate document level events (ie calling startDocument or endDocument).
     */
    public static final String JAXB_FRAGMENT =
        "jaxb.fragment";

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into the specified
     * <tt>javax.xml.transform.Result</tt>.
     *
     * <p>
     * All JAXB Providers must at least support
     * {@link javax.xml.transform.dom.DOMResult},
     * {@link javax.xml.transform.sax.SAXResult}, and
     * {@link javax.xml.transform.stream.StreamResult}. It can
     * support other derived classes of <tt>Result</tt> as well.
     *
     * @param jaxbElement
     *      The root of content tree to be marshalled.
     * @param result
     *      XML will be sent to this Result
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public void marshal( Object jaxbElement, javax.xml.transform.Result result )
        throws JAXBException;
```

```

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into an output stream.
     *
     * @param jaxbElement
     *      The root of content tree to be marshalled.
     * @param os
     *      XML will be added to this stream.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public void marshal( Object jaxbElement, java.io.OutputStream os )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into a file.
     *
     * @param jaxbElement
     *      The root of content tree to be marshalled.
     * @param output
     *      File to be written. If this file already exists, it will be overwritten.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     * @since JAXB2.1
     */
    public void marshal( Object jaxbElement, File output )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into a Writer.
     *
     * @param jaxbElement
     *      The root of content tree to be marshalled.
     * @param writer
     *      XML will be sent to this writer.
```

```
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public void marshal( Object jaxbElement, java.io.Writer writer )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into SAX2 events.
     *
     * @param jaxbElement
     *      The root of content tree to be marshalled.
     * @param handler
     *      XML will be sent to this handler as SAX2 events.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public void marshal( Object jaxbElement, org.xml.sax.ContentHandler handler )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into a DOM tree.
     *
     * @param jaxbElement
     *      The content tree to be marshalled.
     * @param node
     *      DOM nodes will be added as children of this node.
     *      This parameter must be a Node that accepts children
     *      ({@link org.w3c.dom.Document},
     *      {@link  org.w3c.dom.DocumentFragment}, or
     *      {@link  org.w3c.dom.Element})
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
```

```
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>jaxbElement</tt> (or any
     *      object reachable from <tt>jaxbElement</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     */
    public void marshal( Object jaxbElement, org.w3c.dom.Node node )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into a
     * {@link javax.xml.stream.XMLStreamWriter}.
     *
     * @param jaxbElement
     *      The content tree to be marshalled.
     * @param writer
     *      XML will be sent to this writer.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     * @since JAXB 2.0
     */
    public void marshal( Object jaxbElement, javax.xml.stream.XMLStreamWriter writer )
        throws JAXBException;

    /**
     * Marshal the content tree rooted at <tt>jaxbElement</tt> into a
     * {@link javax.xml.stream.XMLEventWriter}.
     *
     * @param jaxbElement
     *      The content tree rooted at jaxbElement to be marshalled.
     * @param writer
     *      XML will be sent to this writer.
     *
     * @throws JAXBException
     *      If any unexpected problem occurs during the marshalling.
     * @throws MarshalException
     *      If the {@link ValidationEventHandler ValidationEventHandler}
     *      returns false from its <tt>handleEvent</tt> method or the
     *      <tt>Marshaller</tt> is unable to marshal <tt>obj</tt> (or any
     *      object reachable from <tt>obj</tt>).  See <a href="#elementMarshalling">
     *      Marshalling a JAXB element</a>.
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
```

```
     * @since JAXB 2.0
     */
    public void marshal( Object jaxbElement, javax.xml.stream.XMLEventWriter writer )
        throws JAXBException;

    /**
     * Get a DOM tree view of the content tree(Optional).
     *
     * If the returned DOM tree is updated, these changes are also
     * visible in the content tree.
     * Use {@link #marshal(Object, org.w3c.dom.Node)} to force
     * a deep copy of the content tree to a DOM representation.
     *
     * @param contentTree - JAXB Java representation of XML content
     *
     * @return the DOM tree view of the contentTree
     *
     * @throws UnsupportedOperationException
     *      If the JAXB provider implementation does not support a
     *      DOM view of the content tree
     *
     * @throws IllegalArgumentException
     *      If any of the method parameters are null
     *
     * @throws JAXBException
     *      If any unexpected problem occurs
     *
     */
    public org.w3c.dom.Node getNode( java.lang.Object contentTree )
        throws JAXBException;

    /**
     * Set the particular property in the underlying implementation of
     * <tt>Marshaller</tt>.  This method can only be used to set one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to set an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to be set. This value can either
     *              be specified using one of the constant fields or a user
     *              supplied string.
     * @param value the value of the property to be set
     *
     * @throws PropertyException when there is an error processing the given
     *                           property or value
     * @throws IllegalArgumentException
     *      If the name parameter is null
     */
    public void setProperty( String name, Object value )
        throws PropertyException;

    /**
```

```
     * Get the particular property in the underlying implementation of
     * <tt>Marshaller</tt>.  This method can only be used to get one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to get an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to retrieve
     * @return the value of the requested property
     *
     * @throws PropertyException
     *      when there is an error retrieving the given property or value
     *      property name
     * @throws IllegalArgumentException
     *      If the name parameter is null
     */
    public Object getProperty( String name ) throws PropertyException;

    /**
     * Allow an application to register a validation event handler.
     * <p>
     * The validation event handler will be called by the JAXB Provider if any
     * validation errors are encountered during calls to any of the marshal
     * API's.  If the client application does not register a validation event
     * handler before invoking one of the marshal methods, then validation
     * events will be handled by the default event handler which will terminate
     * the marshal operation after the first error or fatal error is encountered.
     * <p>
     * Calling this method with a null parameter will cause the Marshaller
     * to revert back to the default default event handler.
     *
     * @param handler the validation event handler
     * @throws JAXBException if an error was encountered while setting the
     *         event handler
     */
    public void setEventHandler( ValidationEventHandler handler )
        throws JAXBException;

    /**
     * Return the current event handler or the default event handler if one
     * hasn't been set.
     *
     * @return the current ValidationEventHandler or the default event handler
     *         if it hasn't been set
     * @throws JAXBException if an error was encountered while getting the
     *         current event handler
     */
    public ValidationEventHandler getEventHandler()
        throws JAXBException;



    /**
```

```
     * Associates a configured instance of {@link XmlAdapter} with this marshaller.
     *
     * <p>
     * This is a convenience method that invokes <code>setAdapter(adapter.getClass(),adapter);</
       code>.
     *
     * @see #setAdapter(Class,XmlAdapter)
     * @throws IllegalArgumentException
     *      if the adapter parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
     * @since JAXB 2.0
     */
    public void setAdapter( XmlAdapter adapter );

    /**
     * Associates a configured instance of {@link XmlAdapter} with this marshaller.
     *
     * <p>
     * Every marshaller internally maintains a
     * {@link java.util.Map}&lt;{@link Class},{@link XmlAdapter}>,
     * which it uses for marshalling classes whose fields/methods are annotated
     * with {@link javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter}.
     *
     * <p>
     * This method allows applications to use a configured instance of {@link XmlAdapter}.
     * When an instance of an adapter is not given, a marshaller will create
     * one by invoking its default constructor.
     *
     * @param type
     *      The type of the adapter. The specified instance will be used when
     *      {@link javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter#value()}
     *      refers to this type.
     * @param adapter
     *      The instance of the adapter to be used. If null, it will un-register
     *      the current adapter set for this type.
     * @throws IllegalArgumentException
     *      if the type parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
     * @since JAXB 2.0
     */
    public <A extends XmlAdapter> void setAdapter( Class<A> type, A adapter );

    /**
     * Gets the adapter associated with the specified type.
     *
     * This is the reverse operation of the {@link #setAdapter} method.
     *
     * @throws IllegalArgumentException
     *      if the type parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
```

```
     * @since JAXB 2.0
     */
    public <A extends XmlAdapter> A getAdapter( Class<A> type );


    /**
     * <p>Associate a context that enables binary data within an XML document
     * to be transmitted as XML-binary optimized attachment.
     * The attachment is referenced from the XML document content model
     * by content-id URIs(cid) references stored within the xml document.
     *
     * @throws IllegalStateException if attempt to concurrently call this
     *                               method during a marshal operation.
     */
    void setAttachmentMarshaller(AttachmentMarshaller am);

    AttachmentMarshaller getAttachmentMarshaller();

    /**
     * Specify the JAXP 1.3 {@link javax.xml.validation.Schema Schema}
     * object that should be used to validate subsequent marshal operations
     * against.  Passing null into this method will disable validation.
     *
     * <p>
     * This method allows the caller to validate the marshalled XML as it's marshalled.
     *
     * <p>
     * Initially this property is set to <tt>null</tt>.
     *
     * @param schema Schema object to validate marshal operations against or null to disable
         validation
     * @throws UnsupportedOperationException could be thrown if this method is
     *         invoked on an Marshaller created from a JAXBContext referencing
     *         JAXB 1.0 mapped classes
     * @since JAXB2.0
     */
    public void setSchema( Schema schema );

    /**
     * Get the JAXP 1.3 {@link javax.xml.validation.Schema Schema} object
     * being used to perform marshal-time validation.  If there is no
     * Schema set on the marshaller, then this method will return null
     * indicating that marshal-time validation will not be performed.
     *
     * @return the Schema object being used to perform marshal-time
     *      validation or null if not present.
     * @throws UnsupportedOperationException could be thrown if this method is
     *         invoked on an Marshaller created from a JAXBContext referencing
     *         JAXB 1.0 mapped classes
     * @since JAXB2.0
     */
    public Schema getSchema();

```

```
    /**
     * <p/>
     * Register an instance of an implementation of this class with a {@link Marshaller} to externally listen
     * for marshal events.
     * <p/>
     * <p/>
     * This class enables pre and post processing of each marshalled object.
     * The event callbacks are called when marshalling from an instance that maps to an xml element or
     * complex type definition. The event callbacks are not called when marshalling from an instance of a
     * Java datatype that represents a simple type definition.
     * <p/>
     * <p/>
     * External listener is one of two different mechanisms for defining marshal event callbacks.
     * See <a href="Marshaller.html#marshalEventCallback">Marshal Event Callbacks</a> for an overview.
     *
     * @see Marshaller#setListener(Listener)
     * @see Marshaller#getListener()
     * @since JAXB2.0
     */
    public static abstract class Listener {
        /**
         * <p/>
         * Callback method invoked before marshalling from <tt>source</tt> to XML.
         * <p/>
         * <p/>
         * This method is invoked just before marshalling process starts to marshal <tt>source</tt>.
         * Note that if the class of <tt>source</tt> defines its own <tt>beforeMarshal</tt> method,
         * the class specific callback method is invoked just before this method is invoked.
         *
         * @param source instance of JAXB mapped class prior to marshalling from it.
         */
        public void beforeMarshal(Object source) {
        }

        /**
         * <p/>
         * Callback method invoked after marshalling <tt>source</tt> to XML.
         * <p/>
         * <p/>
         * This method is invoked after <tt>source</tt> and all its descendants have been marshalled.
         * Note that if the class of <tt>source</tt> defines its own <tt>afterMarshal</tt> method,
         * the class specific callback method is invoked just before this method is invoked.
         *
         * @param source instance of JAXB mapped class after marshalling it.
         */
        public void afterMarshal(Object source) {
        }
```

```
    }

    /**
     * <p>
     * Register marshal event callback {@link Listener} with this {@link Marshaller}.
     *
     * <p>
     * There is only one Listener per Marshaller. Setting a Listener replaces the previous set
         Listener.
     * One can unregister current Listener by setting listener to <tt>null</tt>.
     *
     * @param listener an instance of a class that implements {@link Listener}
     * @since JAXB2.0
     */
    public void setListener(Listener listener);

    /**
     * <p>Return {@link Listener} registered with this {@link Marshaller}.
     *
     * @return registered {@link Listener} or <code>null</code> if no Listener is registered with
         this Marshaller.
     * @since JAXB2.0
     */
    public Listener getListener();
}
```

2.17 Messages.java

Secuencia 19: javax/xml/bind/Messages.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```java

package javax.xml.bind;

import java.text.MessageFormat;
import java.util.ResourceBundle;

/**
 * Formats error messages.
 */
class Messages
{
    static String format( String property ) {
        return format( property, null );
    }

    static String format( String property, Object arg1 ) {
        return format( property, new Object[]{arg1} );
    }

    static String format( String property, Object arg1, Object arg2 ) {
        return format( property, new Object[]{arg1,arg2} );
    }

    static String format( String property, Object arg1, Object arg2, Object arg3 ) {
        return format( property, new Object[]{arg1,arg2,arg3} );
    }

    // add more if necessary.

    /** Loads a string resource and formats it with specified arguments. */
    static String format( String property, Object[] args ) {
        String text = ResourceBundle.getBundle(Messages.class.getName()).getString(property);
        return MessageFormat.format(text,args);
    }

//
//
// Message resources
//
//
    static final String PROVIDER_NOT_FOUND = // 1 arg
        "ContextFinder.ProviderNotFound";

    static final String COULD_NOT_INSTANTIATE = // 2 args
        "ContextFinder.CouldNotInstantiate";

    static final String CANT_FIND_PROPERTIES_FILE = // 1 arg
        "ContextFinder.CantFindPropertiesFile";

    static final String CANT_MIX_PROVIDERS = // 0 args
        "ContextFinder.CantMixProviders";

    static final String MISSING_PROPERTY = // 2 args
```

```
        "ContextFinder.MissingProperty";

    static final String NO_PACKAGE_IN_CONTEXTPATH = // 0 args
        "ContextFinder.NoPackageInContextPath";

    static final String NAME_VALUE = // 2 args
        "PropertyException.NameValue";

    static final String CONVERTER_MUST_NOT_BE_NULL = // 0 args
        "DatatypeConverter.ConverterMustNotBeNull";

    static final String ILLEGAL_CAST = // 2 args
        "JAXBContext.IllegalCast";
}
```

2.18 NotIdentifiableEvent.java

Secuencia 20: javax/xml/bind/NotIdentifiableEvent.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This event indicates that a problem was encountered resolving an ID/IDREF.
 *
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see Validator
 * @see ValidationEventHandler
```

```
 * @since JAXB1.0
 */
public interface NotIdentifiableEvent extends ValidationEvent {

}
```

2.19 ParseConversionEvent.java

Secuencia 21: javax/xml/bind/ParseConversionEvent.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This event indicates that a problem was encountered while converting a
 * string from the XML data into a value of the target Java data type.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see ValidationEvent
 * @see ValidationEventHandler
 * @see Unmarshaller
 * @since JAXB1.0
 */
public interface ParseConversionEvent extends ValidationEvent {

}
```

2.20 PrintConversionEvent.java

Secuencia 22: javax/xml/bind/PrintConversionEvent.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This event indicates that a problem was encountered while converting data
 * from the Java content tree into its lexical representation.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see ValidationEvent
 * @see ValidationEventHandler
 * @see Marshaller
 * @since JAXB1.0
 */
public interface PrintConversionEvent extends ValidationEvent {

}
```

2.21 PropertyException.java

Secuencia 23: javax/xml/bind/PropertyException.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;



/**
 * This exception indicates that an error was encountered while getting or
 * setting a property.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 * @see Validator
 * @see Unmarshaller
 * @since JAXB1.0
 */
public class PropertyException extends JAXBException {

    /**
     * Construct a PropertyException with the specified detail message.  The
     * errorCode and linkedException will default to null.
     *
     * @param message a description of the exception
     */
    public PropertyException(String message) {
        super(message);
    }

    /**
     * Construct a PropertyException with the specified detail message and
     * vendor specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public PropertyException(String message, String errorCode) {
```

```
        super(message, errorCode);
    }

    /**
     * Construct a PropertyException with a linkedException.  The detail
     * message and vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public PropertyException(Throwable exception) {
        super(exception);
    }

    /**
     * Construct a PropertyException with the specified detail message and
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public PropertyException(String message, Throwable exception) {
        super(message, exception);
    }

    /**
     * Construct a PropertyException with the specified detail message, vendor
     * specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public PropertyException(
        String message,
        String errorCode,
        Throwable exception) {
        super(message, errorCode, exception);
    }

    /**
     * Construct a PropertyException whose message field is set based on the
     * name of the property and value.toString().
     *
     * @param name the name of the property related to this exception
     * @param value the value of the property related to this exception
     */
    public PropertyException(String name, Object value) {
        super( Messages.format( Messages.NAME_VALUE,
                                name,
                                value.toString() ) );
    }


```

```
}
```

2.22 SchemaOutputResolver.java

Secuencia 24: javax/xml/bind/SchemaOutputResolver.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import javax.xml.transform.Result;
import java.io.IOException;

/**
 * Controls where a JAXB implementation puts the generates
 * schema files.
 *
 * <p>
 * An implementation of this abstract class has to be provided by the calling
 * application to generate schemas.
 *
 * <p>
 * This is a class, not an interface so as to allow future versions to evolve
 * without breaking the compatibility.
 *
 * @author
 *     Kohsuke Kawaguchi (kohsuke.kawaguchi@sun.com)
 */
public abstract class SchemaOutputResolver {
    /**
     * Decides where the schema file (of the given namespace URI)
```

```
     * will be written, and return it as a {@link Result} object.
     *
     * <p>
     * This method is called only once for any given namespace.
     * IOW, all the components in one namespace is always written
     * into the same schema document.
     *
     * @param namespaceUri
     *      The namespace URI that the schema declares.
     *      Can be the empty string, but never be null.
     * @param suggestedFileName
     *      A JAXB implementation generates an unique file name (like "schema1.xsd")
     *      for the convenience of the callee. This name can be
     *      used for the file name of the schema, or the callee can just
     *      ignore this name and come up with its own name.
     *      This is just a hint.
     *
     * @return
     *      a {@link Result} object that encapsulates the actual destination
     *      of the schema.
     *
     *      If the {@link Result} object has a system ID, it must be an
     *      absolute system ID. Those system IDs are relativized by the caller and used
     *      for &lt;xs:import> statements.
     *
     *      If the {@link Result} object does not have a system ID, a schema
     *      for the namespace URI is generated but it won't be explicitly
     *      &lt;xs:import>ed from other schemas.
     *
     *      If {@code null} is returned, the schema generation for this
     *      namespace URI will be skipped.
     */
    public abstract Result createOutput( String namespaceUri, String suggestedFileName ) throws
        IOException;
}
```

2.23 TypeConstraintException.java

Secuencia 25: javax/xml/bind/TypeConstraintException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This exception indicates that a violation of a dynamically checked type
 * constraint was detected.
 *
 * <p>
 * This exception can be thrown by the generated setter methods of the schema
 * derived Java content classes.  However, since fail-fast validation is
 * an optional feature for JAXB Providers to support, not all setter methods
 * will throw this exception when a type constraint is violated.
 *
 * <p>
 * If this exception is throw while invoking a fail-fast setter, the value of
 * the property is guaranteed to remain unchanged, as if the setter were never
 * called.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc
     .</li></ul>
 * @see ValidationEvent
 * @since JAXB1.0
 */

public class TypeConstraintException extends java.lang.RuntimeException {

    /**
     * Vendor specific error code
     *
     */
    private String errorCode;

    /**
     * Exception reference
     *
     */
    private volatile Throwable linkedException;

    static final long serialVersionUID = -3059799699420143848L;

    /**
     * Construct a TypeConstraintException with the specified detail message.  The
     * errorCode and linkedException will default to null.
```

```
     *
     * @param message a description of the exception
     */
    public TypeConstraintException(String message) {
        this( message, null, null );
    }

    /**
     * Construct a TypeConstraintException with the specified detail message and vendor
     * specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public TypeConstraintException(String message, String errorCode) {
        this( message, errorCode, null );
    }

    /**
     * Construct a TypeConstraintException with a linkedException.  The detail message and
     * vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public TypeConstraintException(Throwable exception) {
        this( null, null, exception );
    }

    /**
     * Construct a TypeConstraintException with the specified detail message and
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public TypeConstraintException(String message, Throwable exception) {
        this( message, null, exception );
    }

    /**
     * Construct a TypeConstraintException with the specified detail message,
     * vendor specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public TypeConstraintException(String message, String errorCode, Throwable exception) {
        super( message );
        this.errorCode = errorCode;
        this.linkedException = exception;
    }

```

```java
    /**
     * Get the vendor specific error code
     *
     * @return a string specifying the vendor specific error code
     */
    public String getErrorCode() {
        return this.errorCode;
    }

    /**
     * Get the linked exception
     *
     * @return the linked Exception, null if none exists
     */
    public Throwable getLinkedException() {
        return linkedException;
    }

    /**
     * Add a linked Exception.
     *
     * @param exception the linked Exception (A null value is permitted and
     *                  indicates that the linked exception does not exist or
     *                  is unknown).
     */
    public void setLinkedException( Throwable exception ) {
        this.linkedException = exception;
    }

    /**
     * Returns a short description of this TypeConstraintException.
     *
     */
    public String toString() {
        return linkedException == null ?
            super.toString() :
            super.toString() + "\n - with linked exception:\n[" +
                                linkedException.toString()+ "]";
    }

    /**
     * Prints this TypeConstraintException and its stack trace (including the stack trace
     * of the linkedException if it is non-null) to the PrintStream.
     *
     * @param s PrintStream to use for output
     */
    public void printStackTrace( java.io.PrintStream s ) {
        if( linkedException != null ) {
          linkedException.printStackTrace(s);
          s.println("--------------- linked to ------------------");
        }

        super.printStackTrace(s);
```

```
    }

    /**
     * Prints this TypeConstraintException and its stack trace (including the stack trace
     * of the linkedException if it is non-null) to <tt>System.err</tt>.
     *
     */
    public void printStackTrace() {
        printStackTrace(System.err);
    }

}
```

2.24 UnmarshalException.java

Secuencia 26: javax/xml/bind/UnmarshalException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This exception indicates that an error has occurred while performing
 * an unmarshal operation that prevents the JAXB Provider from completing
 * the operation.
 *
 * <p>
 * The <tt>ValidationEventHandler</tt> can cause this exception to be thrown
 * during the unmarshal operations.  See
 * {@link ValidationEventHandler#handleEvent(ValidationEvent)
 * ValidationEventHandler.handleEvent(ValidationEvent)}.
```

```
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see JAXBException
 * @see Unmarshaller
 * @see ValidationEventHandler
 * @since JAXB1.0
 */
public class UnmarshalException extends JAXBException {

    /**
     * Construct an UnmarshalException with the specified detail message.  The
     * errorCode and linkedException will default to null.
     *
     * @param message a description of the exception
     */
    public UnmarshalException( String message ) {
        this( message, null, null );
    }

    /**
     * Construct an UnmarshalException with the specified detail message and vendor
     * specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public UnmarshalException( String message, String errorCode ) {
        this( message, errorCode, null );
    }

    /**
     * Construct an UnmarshalException with a linkedException.  The detail message and
     * vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public UnmarshalException( Throwable exception ) {
        this( null, null, exception );
    }

    /**
     * Construct an UnmarshalException with the specified detail message and
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public UnmarshalException( String message, Throwable exception ) {
        this( message, null, exception );
    }

    /**
     * Construct an UnmarshalException with the specified detail message, vendor
```

```
     * specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public UnmarshalException( String message, String errorCode, Throwable exception ) {
        super( message, errorCode, exception );
    }

}
```

2.25 Unmarshaller.java

Secuencia 27: javax/xml/bind/Unmarshaller.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.attachment.AttachmentUnmarshaller;
import javax.xml.validation.Schema;
import java.io.Reader;

/**
 * The <tt>Unmarshaller</tt> class governs the process of deserializing XML
 * data into newly created Java content trees, optionally validating the XML
 * data as it is unmarshalled.  It provides an overloading of unmarshal methods
 * for many different input kinds.
 *
```

```
* <p>
* Unmarshalling from a File:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*       Object o = u.unmarshal( new File( "nosferatu.xml" ) );
*    </pre>
* </blockquote>
*
*
* <p>
* Unmarshalling from an InputStream:
* <blockquote>
*    <pre>
*       InputStream is = new FileInputStream( "nosferatu.xml" );
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*       Object o = u.unmarshal( is );
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a URL:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*       URL url = new URL( "http://beaker.east/nosferatu.xml" );
*       Object o = u.unmarshal( url );
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a StringBuffer using a
* <tt>javax.xml.transform.stream.StreamSource</tt>:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*       StringBuffer xmlStr = new StringBuffer( "&lt;?xml version=&quot;1.0&quot;?&gt;..." );
*       Object o = u.unmarshal( new StreamSource( new StringReader( xmlStr.toString() ) ) );
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a <tt>org.w3c.dom.Node</tt>:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*
*       DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

```
*       dbf.setNamespaceAware(true);
*       DocumentBuilder db = dbf.newDocumentBuilder();
*       Document doc = db.parse(new File( "nosferatu.xml"));

*       Object o = u.unmarshal( doc );
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a <tt>javax.xml.transform.sax.SAXSource</tt> using a
* client specified validating SAX2.0 parser:
* <blockquote>
*    <pre>
*       // configure a validating SAX2.0 parser (Xerces2)
*       static final String JAXP_SCHEMA_LANGUAGE =
*           "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
*       static final String JAXP_SCHEMA_LOCATION =
*           "http://java.sun.com/xml/jaxp/properties/schemaSource";
*       static final String W3C_XML_SCHEMA =
*           "http://www.w3.org/2001/XMLSchema";
*
*       System.setProperty( "javax.xml.parsers.SAXParserFactory",
*                           "org.apache.xerces.jaxp.SAXParserFactoryImpl" );
*
*       SAXParserFactory spf = SAXParserFactory.newInstance();
*       spf.setNamespaceAware(true);
*       spf.setValidating(true);
*       SAXParser saxParser = spf.newSAXParser();
*
*       try {
*           saxParser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
*           saxParser.setProperty(JAXP_SCHEMA_LOCATION, "http://....");
*       } catch (SAXNotRecognizedException x) {
*           // exception handling omitted
*       }
*
*       XMLReader xmlReader = saxParser.getXMLReader();
*       SAXSource source =
*           new SAXSource( xmlReader, new InputSource( "http://..." ) );
*
*       // Setup JAXB to unmarshal
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*       ValidationEventCollector vec = new ValidationEventCollector();
*       u.setEventHandler( vec );
*
*       // turn off the JAXB provider's default validation mechanism to
*       // avoid duplicate validation
*       u.setValidating( false )
*
*       // unmarshal
*       Object o = u.unmarshal( source );
*
```

```
*       // check for events
*       if( vec.hasEvents() ) {
*          // iterate over events
*       }
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a StAX XMLStreamReader:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*
*       javax.xml.stream.XMLStreamReader xmlStreamReader =
*           javax.xml.stream.XMLInputFactory().newInstance().createXMLStreamReader( ... );
*
*       Object o = u.unmarshal( xmlStreamReader );
*    </pre>
* </blockquote>
*
* <p>
* Unmarshalling from a StAX XMLEventReader:
* <blockquote>
*    <pre>
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
*       Unmarshaller u = jc.createUnmarshaller();
*
*       javax.xml.stream.XMLEventReader xmlEventReader =
*           javax.xml.stream.XMLInputFactory().newInstance().createXMLEventReader( ... );
*
*       Object o = u.unmarshal( xmlEventReader );
*    </pre>
* </blockquote>
*
* <p>
* <a name="unmarshalEx"></a>
* <b>Unmarshalling XML Data</b><br>
* <blockquote>
* Unmarshalling can deserialize XML data that represents either an entire XML document
* or a subtree of an XML document. Typically, it is sufficient to use the
* unmarshalling methods described by
* <a href="#unmarshalGlobal">Unmarshal root element that is declared globally</a>.
* These unmarshal methods utilize {@link JAXBContext}'s mapping of global XML element
* declarations and type definitions to JAXB mapped classes to initiate the
* unmarshalling of the root element of  XML data.  When the {@link JAXBContext}'s
* mappings are not sufficient to unmarshal the root element of XML data,
* the application can assist the unmarshalling process by using the
* <a href="#unmarshalByDeclaredType">unmarshal by declaredType methods</a>.
* These methods are useful for unmarshalling XML data where
* the root element corresponds to a local element declaration in the schema.
* </blockquote>
*
```

```
* <blockquote>
* An unmarshal method never returns null. If the unmarshal process is unable to unmarshal
* the root of XML content to a JAXB mapped object, a fatal error is reported that
* terminates processing by throwing JAXBException.
* </blockquote>
*
* <p>
* <a name="unmarshalGlobal"></a>
* <b>Unmarshal a root element that is globally declared</b><br>
* <blockquote>
* The unmarshal methods that do not have an <tt>declaredType</tt> parameter use
* {@link JAXBContext} to unmarshal the root element of an XML data. The {@link JAXBContext}
* instance is the one that was used to create this <tt>Unmarshaller</tt>. The {@link JAXBContext}
* instance maintains a mapping of globally declared XML element and type definition names to
* JAXB mapped classes. The unmarshal method checks if {@link JAXBContext} has a mapping
* from the root element's XML name and/or <tt>@xsi:type</tt> to a JAXB mapped class.  If it does, it umarshalls the
* XML data using the appropriate JAXB mapped class. Note that when the root element name is unknown and the root
* element has an <tt>@xsi:type</tt>, the XML data is unmarshalled
* using that JAXB mapped class as the value of a {@link JAXBElement}.
* When the {@link JAXBContext} object does not have a mapping for the root element's name
* nor its <tt>@xsi:type</tt>, if it exists,
* then the unmarshal operation will abort immediately by throwing a {@link UnmarshalException
* UnmarshalException}. This exception scenario can be worked around by using the unmarshal by
* declaredType methods described in the next subsection.
* </blockquote>
*
* <p>
* <a name="unmarshalByDeclaredType"></a>
* <b>Unmarshal by Declared Type</b><br>
* <blockquote>
* The unmarshal methods with a <code>declaredType</code> parameter enable an
* application to deserialize a root element of XML data, even when
* there is no mapping in {@link JAXBContext} of the root element's XML name.
* The unmarshaller unmarshals the root element using the application provided
* mapping specified as the <tt>declaredType</tt> parameter.
* Note that even when the root element's element name is mapped by {@link JAXBContext},
* the <code>declaredType</code> parameter overrides that mapping for
* deserializing the root element when using these unmarshal methods.
* Additionally, when the root element of XML data has an <tt>xsi:type</tt> attribute and
* that attribute's value references a type definition that is mapped
* to a JAXB mapped class by {@link JAXBContext}, that the root
* element's <tt>xsi:type</tt> attribute takes
* precedence over the unmarshal methods <tt>declaredType</tt> parameter.
* These methods always return a <tt>JAXBElement&lt;declaredType></tt>
* instance. The table below shows how the properties of the returned JAXBElement
* instance are set.
*
* <a name="unmarshalDeclaredTypeReturn"></a>
* <p>
*   <table border="2" rules="all" cellpadding="4">
*   <thead>
```

```
*     <tr>
*       <th align="center" colspan="2">
*       Unmarshal By Declared Type returned JAXBElement
*       </tr>
*     <tr>
*       <th>JAXBElement Property</th>
*       <th>Value</th>
*     </tr>
*     <tr>
*       <td>name</td>
*       <td><code>xml element name</code></td>
*     </tr>
*   </thead>
*   <tbody>
*     <tr>
*       <td>value</td>
*       <td><code>instanceof declaredType</code></td>
*     </tr>
*     <tr>
*       <td>declaredType</td>
*       <td>unmarshal method <code>declaredType</code> parameter</td>
*     </tr>
*     <tr>
*       <td>scope</td>
*       <td><code>null</code> <i>(actual scope is unknown)</i></td>
*     </tr>
*   </tbody>
*  </table>
* </blockquote>
*
* <p>
* The following is an example of
* <a href="#unmarshalByDeclaredType">unmarshal by declaredType method</a>.
* <p>
* Unmarshal by declaredType from a <tt>org.w3c.dom.Node</tt>:
* <blockquote>
*    <pre>
*       Schema fragment for example
*       &lt;xs:schema>
*          &lt;xs:complexType name="FooType">...&lt;\xs:complexType>
*          &lt;!-- global element declaration "PurchaseOrder" -->
*          &lt;xs:element name="PurchaseOrder">
*              &lt;xs:complexType>
*                 &lt;xs:sequence>
*                    &lt;!-- local element declaration "foo" -->
*                    &lt;xs:element name="foo" type="FooType"/>
*                    ...
*                 &lt;/xs:sequence>
*              &lt;/xs:complexType>
*          &lt;/xs:element>
*       &lt;/xs:schema>
*
*       JAXBContext jc = JAXBContext.newInstance( "com.acme.foo" );
```

```
*       Unmarshaller u = jc.createUnmarshaller();
*
*       DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
*       dbf.setNamespaceAware(true);
*       DocumentBuilder db = dbf.newDocumentBuilder();
*       Document doc = db.parse(new File( "nosferatu.xml"));
*       Element  fooSubtree = ...; // traverse DOM till reach xml element foo, constrained by a
*                                  // local element declaration in schema.
*
*       // FooType is the JAXB mapping of the type of local element declaration foo.
*       JAXBElement&lt;FooType> foo = u.unmarshal( fooSubtree, FooType.class);
*    </pre>
* </blockquote>
*
* <p>
* <b>Support for SAX2.0 Compliant Parsers</b><br>
* <blockquote>
* A client application has the ability to select the SAX2.0 compliant parser
* of their choice.  If a SAX parser is not selected, then the JAXB Provider's
* default parser will be used.  Even though the JAXB Provider's default parser
* is not required to be SAX2.0 compliant, all providers are required to allow
* a client application to specify their own SAX2.0 parser.  Some providers may
* require the client application to specify the SAX2.0 parser at schema compile
* time. See {@link #unmarshal(javax.xml.transform.Source) unmarshal(Source)}
* for more detail.
* </blockquote>
*
* <p>
* <b>Validation and Well-Formedness</b><br>
* <blockquote>
* <p>
* A client application can enable or disable JAXP 1.3 validation
* mechanism via the <tt>setSchema(javax.xml.validation.Schema)</tt> API.
* Sophisticated clients can specify their own validating SAX 2.0 compliant
* parser and bypass the JAXP 1.3 validation mechanism using the
* {@link #unmarshal(javax.xml.transform.Source) unmarshal(Source)}  API.
*
* <p>
* Since unmarshalling invalid XML content is defined in JAXB 2.0,
* the Unmarshaller default validation event handler was made more lenient
* than in JAXB 1.0.  When schema-derived code generated
* by JAXB 1.0 binding compiler is registered with {@link JAXBContext},
* the default unmarshal validation handler is
* {@link javax.xml.bind.helpers.DefaultValidationEventHandler} and it
* terminates the marshal  operation after encountering either a fatal error or an error.
* For a JAXB 2.0 client application, there is no explicitly defined default
* validation handler and the default event handling only
* terminates the unmarshal operation after encountering a fatal error.
*
* </blockquote>
*
* <p>
* <a name="supportedProps"></a>
```

```
 * <b>Supported Properties</b><br>
 * <blockquote>
 * <p>
 * There currently are not any properties required to be supported by all
 * JAXB Providers on Unmarshaller.  However, some providers may support
 * their own set of provider specific properties.
 * </blockquote>
 *
 * <p>
 * <a name="unmarshalEventCallback"></a>
 * <b>Unmarshal Event Callbacks</b><br>
 * <blockquote>
 * The {@link Unmarshaller} provides two styles of callback mechanisms
 * that allow application specific processing during key points in the
 * unmarshalling process.  In 'class defined' event callbacks, application
 * specific code placed in JAXB mapped classes is triggered during
 * unmarshalling.  'External listeners' allow for centralized processing
 * of unmarshal events in one callback method rather than by type event callbacks.
 * <p>
 * 'Class defined' event callback methods allow any JAXB mapped class to specify
 * its own specific callback methods by defining methods with the following method signature:
 * <blockquote>
 * <pre>
 *   // This method is called immediately after the object is created and before the unmarshalling of this
 *   // object begins. The callback provides an opportunity to initialize JavaBean properties prior to unmarshalling.
 *   void beforeUnmarshal(Unmarshaller, Object parent);
 *
 *   //This method is called after all the properties (except IDREF) are unmarshalled for this object,
 *   //but before this object is set to the parent object.
 *   void afterUnmarshal(Unmarshaller, Object parent);
 * </pre>
 * </blockquote>
 * The class defined callback methods should be used when the callback method requires
 * access to non-public methods and/or fields of the class.
 * <p>
 * The external listener callback mechanism enables the registration of a {@link Listener}
 * instance with an {@link Unmarshaller#setListener(Listener)}. The external listener receives all callback events,
 * allowing for more centralized processing than per class defined callback methods.  The external listener
 * receives events when unmarshalling proces is marshalling to a JAXB element or to JAXB mapped class.
 * <p>
 * The 'class defined' and external listener event callback methods are independent of each other,
 * both can be called for one event.  The invocation ordering when both listener callback methods exist is
 * defined in {@link Listener#beforeUnmarshal(Object, Object)} and {@link Listener#afterUnmarshal(Object, Object)}.
* <p>
 * An event callback method throwing an exception terminates the current unmarshal process.
```

```
 *
 * </blockquote>
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 * @see Marshaller
 * @see Validator
 * @since JAXB1.0
 */
public interface Unmarshaller {

    /**
     * Unmarshal XML data from the specified file and return the resulting
     * content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param f the file to unmarshal XML data from
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the file parameter is null
     */
    public Object unmarshal( java.io.File f ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified InputStream and return the
     * resulting content tree.  Validation event location information may
     * be incomplete when using this form of the unmarshal API.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param is the InputStream to unmarshal XML data from
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
```

```
     *      If the InputStream parameter is null
     */
    public Object unmarshal( java.io.InputStream is ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified Reader and return the
     * resulting content tree.  Validation event location information may
     * be incomplete when using this form of the unmarshal API,
     * because a Reader does not provide the system ID.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param reader the Reader to unmarshal XML data from
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the InputStream parameter is null
     * @since JAXB2.0
     */
    public Object unmarshal( Reader reader ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified URL and return the resulting
     * content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param url the url to unmarshal XML data from
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the URL parameter is null
     */
    public Object unmarshal( java.net.URL url ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified SAX InputSource and return the
```

```
     * resulting content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param source the input source to unmarshal XML data from
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the InputSource parameter is null
     */
    public Object unmarshal( org.xml.sax.InputSource source ) throws JAXBException;

    /**
     * Unmarshal global XML data from the specified DOM tree and return the resulting
     * content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * @param node
     *      the document/element to unmarshal XML data from.
     *      The caller must support at least Document and Element.
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the Node parameter is null
     * @see #unmarshal(org.w3c.dom.Node, Class)
     */
    public Object unmarshal( org.w3c.dom.Node node ) throws JAXBException;

    /**
     * Unmarshal XML data by JAXB mapped <tt>declaredType</tt>
     * and return the resulting content tree.
     *
     * <p>
     * Implements <a href="#unmarshalByDeclaredType">Unmarshal by Declared Type</a>
     *
     * @param node
```

```
     *      the document/element to unmarshal XML data from.
     *      The caller must support at least Document and Element.
     * @param declaredType
     *      appropriate JAXB mapped class to hold <tt>node</tt>'s XML data.
     *
     * @return <a href="#unmarshalDeclaredTypeReturn">JAXB Element</a> representation of <tt>node</tt>
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If any parameter is null
     * @since JAXB2.0
     */
    public <T> JAXBElement<T> unmarshal( org.w3c.dom.Node node, Class<T> declaredType ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified XML Source and return the
     * resulting content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * <p>
     * <a name="saxParserPlugable"></a>
     * <b>SAX 2.0 Parser Pluggability</b>
     * <p>
     * A client application can choose not to use the default parser mechanism
     * supplied with their JAXB provider.  Any SAX 2.0 compliant parser can be
     * substituted for the JAXB provider's default mechanism.  To do so, the
     * client application must properly configure a <tt>SAXSource</tt> containing
     * an <tt>XMLReader</tt> implemented by the SAX 2.0 parser provider.  If the
     * <tt>XMLReader</tt> has an <tt>org.xml.sax.ErrorHandler</tt> registered
     * on it, it will be replaced by the JAXB Provider so that validation errors
     * can be reported via the <tt>ValidationEventHandler</tt> mechanism of
     * JAXB.  If the <tt>SAXSource</tt> does not contain an <tt>XMLReader</tt>,
     * then the JAXB provider's default parser mechanism will be used.
     * <p>
     * This parser replacement mechanism can also be used to replace the JAXB
     * provider's unmarshal-time validation engine.  The client application
     * must properly configure their SAX 2.0 compliant parser to perform
     * validation (as shown in the example above).  Any <tt>SAXParserExceptions
     * </tt> encountered by the parser during the unmarshal operation will be
     * processed by the JAXB provider and converted into JAXB
     * <tt>ValidationEvent</tt> objects which will be reported back to the
     * client via the <tt>ValidationEventHandler</tt> registered with the
     * <tt>Unmarshaller</tt>.  <i>Note:</i> specifying a substitute validating
```

```
     * SAX 2.0 parser for unmarshalling does not necessarily replace the
     * validation engine used by the JAXB provider for performing on-demand
     * validation.
     * <p>
     * The only way for a client application to specify an alternate parser
     * mechanism to be used during unmarshal is via the
     * <tt>unmarshal(SAXSource)</tt> API.  All other forms of the unmarshal
     * method (File, URL, Node, etc) will use the JAXB provider's default
     * parser and validator mechanisms.
     *
     * @param source the XML Source to unmarshal XML data from (providers are
     *        only required to support SAXSource, DOMSource, and StreamSource)
     * @return the newly created root object of the java content tree
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the Source parameter is null
     * @see #unmarshal(javax.xml.transform.Source, Class)
     */
    public Object unmarshal( javax.xml.transform.Source source )
        throws JAXBException;


    /**
     * Unmarshal XML data from the specified XML Source by <tt>declaredType</tt> and return the
     * resulting content tree.
     *
     * <p>
     * Implements <a href="#unmarshalByDeclaredType">Unmarshal by Declared Type</a>
     *
     * <p>
     * See <a href="#saxParserPlugable">SAX 2.0 Parser Pluggability</a>
     *
     * @param source the XML Source to unmarshal XML data from (providers are
     *        only required to support SAXSource, DOMSource, and StreamSource)
     * @param declaredType
     *      appropriate JAXB mapped class to hold <tt>source</tt>'s xml root element
     * @return Java content rooted by <a href="#unmarshalDeclaredTypeReturn">JAXB Element</a>
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
```

```
     *      If any parameter is null
     * @since JAXB2.0
     */
    public <T> JAXBElement<T> unmarshal( javax.xml.transform.Source source, Class<T> declaredType )
        throws JAXBException;

    /**
     * Unmarshal XML data from the specified pull parser and return the
     * resulting content tree.
     *
     * <p>
     * Implements <a href="#unmarshalGlobal">Unmarshal Global Root Element</a>.
     *
     * <p>
     * This method assumes that the parser is on a START_DOCUMENT or
     * START_ELEMENT event.  Unmarshalling will be done from this
     * start event to the corresponding end event.  If this method
     * returns successfully, the <tt>reader</tt> will be pointing at
     * the token right after the end event.
     *
     * @param reader
     *      The parser to be read.
     * @return
     *      the newly created root object of the java content tree.
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the <tt>reader</tt> parameter is null
     * @throws IllegalStateException
     *      If <tt>reader</tt> is not pointing to a START_DOCUMENT or
     *      START_ELEMENT  event.
     * @since JAXB2.0
     * @see #unmarshal(javax.xml.stream.XMLStreamReader, Class)
     */
    public Object unmarshal( javax.xml.stream.XMLStreamReader reader )
        throws JAXBException;

    /**
     * Unmarshal root element to JAXB mapped <tt>declaredType</tt>
     * and return the resulting content tree.
     *
     * <p>
     * This method implements <a href="#unmarshalByDeclaredType">unmarshal by declaredType</a>.
     * <p>
     * This method assumes that the parser is on a START_DOCUMENT or
     * START_ELEMENT event. Unmarshalling will be done from this
     * start event to the corresponding end event.  If this method
```

```
     * returns successfully, the <tt>reader</tt> will be pointing at
     * the token right after the end event.
     *
     * @param reader
     *      The parser to be read.
     * @param declaredType
     *      appropriate JAXB mapped class to hold <tt>reader</tt>'s START_ELEMENT XML data.
     *
     * @return   content tree rooted by <a href="#unmarshalDeclaredTypeReturn">JAXB Element
         representation</a>
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If any parameter is null
     * @since JAXB2.0
     */
    public <T> JAXBElement<T> unmarshal( javax.xml.stream.XMLStreamReader reader, Class<T>
        declaredType ) throws JAXBException;

    /**
     * Unmarshal XML data from the specified pull parser and return the
     * resulting content tree.
     *
     * <p>
     * This method is an <a href="#unmarshalGlobal">Unmarshal Global Root method</a>.
     *
     * <p>
     * This method assumes that the parser is on a START_DOCUMENT or
     * START_ELEMENT event.  Unmarshalling will be done from this
     * start event to the corresponding end event.  If this method
     * returns successfully, the <tt>reader</tt> will be pointing at
     * the token right after the end event.
     *
     * @param reader
     *      The parser to be read.
     * @return
     *      the newly created root object of the java content tree.
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If the <tt>reader</tt> parameter is null
```

```
     * @throws IllegalStateException
     *      If <tt>reader</tt> is not pointing to a START_DOCUMENT or
     *      START_ELEMENT event.
     * @since JAXB2.0
     * @see #unmarshal(javax.xml.stream.XMLEventReader, Class)
     */
    public Object unmarshal( javax.xml.stream.XMLEventReader reader )
        throws JAXBException;

    /**
     * Unmarshal root element to JAXB mapped <tt>declaredType</tt>
     * and return the resulting content tree.
     *
     * <p>
     * This method implements <a href="#unmarshalByDeclaredType">unmarshal by declaredType</a>.
     *
     * <p>
     * This method assumes that the parser is on a START_DOCUMENT or
     * START_ELEMENT event. Unmarshalling will be done from this
     * start event to the corresponding end event.  If this method
     * returns successfully, the <tt>reader</tt> will be pointing at
     * the token right after the end event.
     *
     * @param reader
     *      The parser to be read.
     * @param declaredType
     *      appropriate JAXB mapped class to hold <tt>reader</tt>'s START_ELEMENT XML data.
     *
     * @return   content tree rooted by <a href="#unmarshalDeclaredTypeReturn">JAXB Element
         representation</a>
     *
     * @throws JAXBException
     *     If any unexpected errors occur while unmarshalling
     * @throws UnmarshalException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Unmarshaller</tt> is unable to perform the XML to Java
     *     binding.  See <a href="#unmarshalEx">Unmarshalling XML Data</a>
     * @throws IllegalArgumentException
     *      If any parameter is null
     * @since JAXB2.0
     */
    public <T> JAXBElement<T> unmarshal( javax.xml.stream.XMLEventReader reader, Class<T>
        declaredType ) throws JAXBException;

    /**
     * Get an unmarshaller handler object that can be used as a component in
     * an XML pipeline.
     *
     * <p>
     * The JAXB Provider can return the same handler object for multiple
     * invocations of this method. In other words, this method does not
     * necessarily create a new instance of <tt>UnmarshallerHandler</tt>. If the
```

```
     * application needs to use more than one <tt>UnmarshallerHandler</tt>, it
     * should create more than one <tt>Unmarshaller</tt>.
     *
     * @return the unmarshaller handler object
     * @see UnmarshallerHandler
     */
    public UnmarshallerHandler getUnmarshallerHandler();

    /**
     * Specifies whether or not the default validation mechanism of the
     * <tt>Unmarshaller</tt> should validate during unmarshal operations.
     * By default, the <tt>Unmarshaller</tt> does not validate.
     * <p>
     * This method may only be invoked before or after calling one of the
     * unmarshal methods.
     * <p>
     * This method only controls the JAXB Provider's default unmarshal-time
     * validation mechanism - it has no impact on clients that specify their
     * own validating SAX 2.0 compliant parser.  Clients that specify their
     * own unmarshal-time validation mechanism may wish to turn off the JAXB
     * Provider's default validation mechanism via this API to avoid "double
     * validation".
     * <p>
     * This method is deprecated as of JAXB 2.0 - please use the new
     * {@link #setSchema(javax.xml.validation.Schema)} API.
     *
     * @param validating true if the Unmarshaller should validate during
     *        unmarshal, false otherwise
     * @throws JAXBException if an error occurred while enabling or disabling
     *         validation at unmarshal time
     * @throws UnsupportedOperationException could be thrown if this method is
     *         invoked on an Unmarshaller created from a JAXBContext referencing
     *         JAXB 2.0 mapped classes
     * @deprecated since JAXB2.0, please see {@link #setSchema(javax.xml.validation.Schema)}
     */
    public void setValidating( boolean validating )
        throws JAXBException;

    /**
     * Indicates whether or not the <tt>Unmarshaller</tt> is configured to
     * validate during unmarshal operations.
     * <p>
     * This API returns the state of the JAXB Provider's default unmarshal-time
     * validation mechanism.
     * <p>
     * This method is deprecated as of JAXB 2.0 - please use the new
     * {@link #getSchema()} API.
     *
     * @return true if the Unmarshaller is configured to validate during
     *         unmarshal operations, false otherwise
     * @throws JAXBException if an error occurs while retrieving the validating
     *         flag
     * @throws UnsupportedOperationException could be thrown if this method is
```

```
     *         invoked on an Unmarshaller created from a JAXBContext referencing
     *         JAXB 2.0 mapped classes
     * @deprecated since JAXB2.0, please see {@link #getSchema()}
     */
    public boolean isValidating()
        throws JAXBException;

    /**
     * Allow an application to register a <tt>ValidationEventHandler</tt>.
     * <p>
     * The <tt>ValidationEventHandler</tt> will be called by the JAXB Provider
     * if any validation errors are encountered during calls to any of the
     * unmarshal methods.  If the client application does not register a
     * <tt>ValidationEventHandler</tt> before invoking the unmarshal methods,
     * then <tt>ValidationEvents</tt> will be handled by the default event
     * handler which will terminate the unmarshal operation after the first
     * error or fatal error is encountered.
     * <p>
     * Calling this method with a null parameter will cause the Unmarshaller
     * to revert back to the default event handler.
     *
     * @param handler the validation event handler
     * @throws JAXBException if an error was encountered while setting the
     *         event handler
     */
    public void setEventHandler( ValidationEventHandler handler )
        throws JAXBException;

    /**
     * Return the current event handler or the default event handler if one
     * hasn't been set.
     *
     * @return the current ValidationEventHandler or the default event handler
     *         if it hasn't been set
     * @throws JAXBException if an error was encountered while getting the
     *         current event handler
     */
    public ValidationEventHandler getEventHandler()
        throws JAXBException;

    /**
     * Set the particular property in the underlying implementation of
     * <tt>Unmarshaller</tt>.  This method can only be used to set one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to set an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to be set. This value can either
     *              be specified using one of the constant fields or a user
     *              supplied string.
     * @param value the value of the property to be set
     *
```

```
     * @throws PropertyException when there is an error processing the given
     *                           property or value
     * @throws IllegalArgumentException
     *      If the name parameter is null
     */
    public void setProperty( String name, Object value )
        throws PropertyException;

    /**
     * Get the particular property in the underlying implementation of
     * <tt>Unmarshaller</tt>.  This method can only be used to get one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to get an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to retrieve
     * @return the value of the requested property
     *
     * @throws PropertyException
     *      when there is an error retrieving the given property or value
     *      property name
     * @throws IllegalArgumentException
     *      If the name parameter is null
     */
    public Object getProperty( String name ) throws PropertyException;

    /**
     * Specify the JAXP 1.3 {@link javax.xml.validation.Schema Schema}
     * object that should be used to validate subsequent unmarshal operations
     * against.  Passing null into this method will disable validation.
     * <p>
     * This method replaces the deprecated {@link #setValidating(boolean) setValidating(boolean)}
     * API.
     *
     * <p>
     * Initially this property is set to <tt>null</tt>.
     *
     * @param schema Schema object to validate unmarshal operations against or null to disable
         validation
     * @throws UnsupportedOperationException could be thrown if this method is
     *         invoked on an Unmarshaller created from a JAXBContext referencing
     *         JAXB 1.0 mapped classes
     * @since JAXB2.0
     */
    public void setSchema( javax.xml.validation.Schema schema );

    /**
     * Get the JAXP 1.3 {@link javax.xml.validation.Schema Schema} object
     * being used to perform unmarshal-time validation.  If there is no
     * Schema set on the unmarshaller, then this method will return null
     * indicating that unmarshal-time validation will not be performed.
     * <p>
```

```
     * This method provides replacement functionality for the deprecated
     * {@link #isValidating()} API as well as access to the Schema object.
     * To determine if the Unmarshaller has validation enabled, simply
     * test the return type for null:
     * <p>
     * <code>
     *   boolean isValidating = u.getSchema()!=null;
     * </code>
     *
     * @return the Schema object being used to perform unmarshal-time
     *      validation or null if not present
     * @throws UnsupportedOperationException could be thrown if this method is
     *         invoked on an Unmarshaller created from a JAXBContext referencing
     *         JAXB 1.0 mapped classes
     * @since JAXB2.0
     */
    public javax.xml.validation.Schema getSchema();

    /**
     * Associates a configured instance of {@link XmlAdapter} with this unmarshaller.
     *
     * <p>
     * This is a convenience method that invokes <code>setAdapter(adapter.getClass(),adapter);</
         code>.
     *
     * @see #setAdapter(Class,XmlAdapter)
     * @throws IllegalArgumentException
     *      if the adapter parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
     * @since JAXB2.0
     */
    public void setAdapter( XmlAdapter adapter );

    /**
     * Associates a configured instance of {@link XmlAdapter} with this unmarshaller.
     *
     * <p>
     * Every unmarshaller internally maintains a
     * {@link java.util.Map}&lt;{@link Class},{@link XmlAdapter}>,
     * which it uses for unmarshalling classes whose fields/methods are annotated
     * with {@link javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter}.
     *
     * <p>
     * This method allows applications to use a configured instance of {@link XmlAdapter}.
     * When an instance of an adapter is not given, an unmarshaller will create
     * one by invoking its default constructor.
     *
     * @param type
     *      The type of the adapter. The specified instance will be used when
     *      {@link javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter#value()}
     *      refers to this type.
     * @param adapter
```

```
     *      The instance of the adapter to be used. If null, it will un-register
     *      the current adapter set for this type.
     * @throws IllegalArgumentException
     *      if the type parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
     * @since JAXB2.0
     */
    public <A extends XmlAdapter> void setAdapter( Class<A> type, A adapter );

    /**
     * Gets the adapter associated with the specified type.
     *
     * This is the reverse operation of the {@link #setAdapter} method.
     *
     * @throws IllegalArgumentException
     *      if the type parameter is null.
     * @throws UnsupportedOperationException
     *      if invoked agains a JAXB 1.0 implementation.
     * @since JAXB2.0
     */
    public <A extends XmlAdapter> A getAdapter( Class<A> type );

    /**
     * <p>Associate a context that resolves cid's, content-id URIs, to
     * binary data passed as attachments.</p>
     * <p/>
     * <p>Unmarshal time validation, enabled via {@link #setSchema(Schema)},
     * must be supported even when unmarshaller is performing XOP processing.
     * </p>
     *
     * @throws IllegalStateException if attempt to concurrently call this
     *                               method during a unmarshal operation.
     */
    void setAttachmentUnmarshaller(AttachmentUnmarshaller au);

    AttachmentUnmarshaller getAttachmentUnmarshaller();

    /**
     * <p/>
     * Register an instance of an implementation of this class with {@link Unmarshaller} to
       externally listen
     * for unmarshal events.
     * <p/>
     * <p/>
     * This class enables pre and post processing of an instance of a JAXB mapped class
     * as XML data is unmarshalled into it. The event callbacks are called when unmarshalling
     * XML content into a JAXBElement instance or a JAXB mapped class that represents a complex
       type definition.
     * The event callbacks are not called when unmarshalling to an instance of a
     * Java datatype that represents a simple type definition.
     * <p/>
     * <p/>
```

```
     * External listener is one of two different mechanisms for defining unmarshal event callbacks.
     * See <a href="Unmarshaller.html#unmarshalEventCallback">Unmarshal Event Callbacks</a> for an overview.
     * <p/>
     * (@link #setListener(Listener)}
     * (@link #getListener()}
     *
     * @since JAXB2.0
     */
    public static abstract class Listener {
        /**
         * <p/>
         * Callback method invoked before unmarshalling into <tt>target</tt>.
         * <p/>
         * <p/>
         * This method is invoked immediately after <tt>target</tt> was created and
         * before the unmarshalling of this object begins. Note that
         * if the class of <tt>target</tt> defines its own <tt>beforeUnmarshal</tt> method,
         * the class specific callback method is invoked before this method is invoked.
         *
         * @param target non-null instance of JAXB mapped class prior to unmarshalling into it.
         * @param parent instance of JAXB mapped class that will eventually reference <tt>target</tt>.
         *               <tt>null</tt> when <tt>target</tt> is root element.
         */
        public void beforeUnmarshal(Object target, Object parent) {
        }

        /**
         * <p/>
         * Callback method invoked after unmarshalling XML data into <tt>target</tt>.
         * <p/>
         * <p/>
         * This method is invoked after all the properties (except IDREF) are unmarshalled into <tt>target</tt>,
         * but before <tt>target</tt> is set into its <tt>parent</tt> object.
         * Note that if the class of <tt>target</tt> defines its own <tt>afterUnmarshal</tt> method,
         * the class specific callback method is invoked before this method is invoked.
         *
         * @param target non-null instance of JAXB mapped class prior to unmarshalling into it.
         * @param parent instance of JAXB mapped class that will reference <tt>target</tt>.
         *               <tt>null</tt> when <tt>target</tt> is root element.
         */
        public void afterUnmarshal(Object target, Object parent) {
        }
    }

    /**
     * <p>
     * Register unmarshal event callback {@link Listener} with this {@link Unmarshaller}.
     *
     * <p>
```

```
     * There is only one Listener per Unmarshaller. Setting a Listener replaces the previous set
         Listener.
     * One can unregister current Listener by setting listener to <tt>null</tt>.
     *
     * @param listener  provides unmarshal event callbacks for this {@link Unmarshaller}
     * @since JAXB2.0
     */
    public void     setListener(Listener listener);

    /**
     * <p>Return {@link Listener} registered with this {@link Unmarshaller}.
     *
     * @return registered {@link Listener} or <code>null</code> if no Listener is registered with
         this Unmarshaller.
     * @since JAXB2.0
     */
    public Listener getListener();
}
```

2.26 UnmarshallerHandler.java

Secuencia 28: javax/xml/bind/UnmarshallerHandler.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

import org.xml.sax.ContentHandler;

/**
 * Unmarshaller implemented as SAX ContentHandler.
```

```
 *
 * <p>
 * Applications can use this interface to use their JAXB provider as a component
 * in an XML pipeline.  For example:
 *
 * <pre>
 *       JAXBContext context = JAXBContext.newInstance( "org.acme.foo" );
 *
 *       Unmarshaller unmarshaller = context.createUnmarshaller();
 *
 *       UnmarshallerHandler unmarshallerHandler = unmarshaller.getUnmarshallerHandler();
 *
 *       SAXParserFactory spf = SAXParserFactory.newInstance();
 *       spf.setNamespaceAware( true );
 *
 *       XMLReader xmlReader = spf.newSAXParser().getXMLReader();
 *       xmlReader.setContentHandler( unmarshallerHandler );
 *       xmlReader.parse(new InputSource( new FileInputStream( XML_FILE ) ) );
 *
 *       MyObject myObject= (MyObject)unmarshallerHandler.getResult();
 * </pre>
 *
 * <p>
 * This interface is reusable: even if the user fails to unmarshal
 * an object, s/he can still start a new round of unmarshalling.
 *
 * @author <ul><li>Kohsuke KAWAGUCHI, Sun Microsystems, Inc.</li></ul>
 * @see Unmarshaller#getUnmarshallerHandler()
 * @since JAXB1.0
 */
public interface UnmarshallerHandler extends ContentHandler
{
    /**
     * Obtains the unmarshalled result.
     *
     * This method can be called only after this handler
     * receives the endDocument SAX event.
     *
     * @exception IllegalStateException
     *      if this method is called before this handler
     *      receives the endDocument event.
     *
     * @exception JAXBException
     *      if there is any unmarshalling error.
     *      Note that the implementation is allowed to throw SAXException
     *      during the parsing when it finds an error.
     *
     * @return
     *      always return a non-null valid object which was unmarshalled.
     */
    Object getResult() throws JAXBException, IllegalStateException;
}
```

2.27 ValidationEvent.java

Secuencia 29: javax/xml/bind/ValidationEvent.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * This event indicates that a problem was encountered while validating the
 * incoming XML data during an unmarshal operation, while performing
 * on-demand validation of the Java content tree, or while marshalling the
 * Java content tree back to XML data.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see Validator
 * @see ValidationEventHandler
 * @since JAXB1.0
 */
public interface ValidationEvent {

    /**
     * Conditions that are not errors or fatal errors as defined by the
     * XML 1.0 recommendation
     */
    public static final int WARNING     = 0;

    /**
     * Conditions that correspond to the definition of "error" in section
     * 1.2 of the W3C XML 1.0 Recommendation
```

```
     */
    public static final int ERROR       = 1;

    /**
     * Conditions that correspond to the definition of "fatal error" in section
     * 1.2 of the W3C XML 1.0 Recommendation
     */
    public static final int FATAL_ERROR = 2;

    /**
     * Retrieve the severity code for this warning/error.
     *
     * <p>
     * Must be one of <tt>ValidationError.WARNING</tt>,
     * <tt>ValidationError.ERROR</tt>, or <tt>ValidationError.FATAL_ERROR</tt>.
     *
     * @return the severity code for this warning/error
     */
    public int getSeverity();

    /**
     * Retrieve the text message for this warning/error.
     *
     * @return the text message for this warning/error or null if one wasn't set
     */
    public String getMessage();

    /**
     * Retrieve the linked exception for this warning/error.
     *
     * @return the linked exception for this warning/error or null if one
     *         wasn't set
     */
    public Throwable getLinkedException();

    /**
     * Retrieve the locator for this warning/error.
     *
     * @return the locator that indicates where the warning/error occurred
     */
    public ValidationEventLocator getLocator();

}
```

2.28 ValidationEventHandler.java

Secuencia 30: javax/xml/bind/ValidationEventHandler.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * A basic event handler interface for validation errors.
 *
 * <p>
 * If an application needs to implement customized event handling, it must
 * implement this interface and then register it with either the
 * {@link Unmarshaller#setEventHandler(ValidationEventHandler) Unmarshaller},
 * the {@link Validator#setEventHandler(ValidationEventHandler) Validator}, or
 * the {@link Marshaller#setEventHandler(ValidationEventHandler) Marshaller}.
 * The JAXB Provider will then report validation errors and warnings encountered
 * during the unmarshal, marshal, and validate operations to these event
 * handlers.
 *
 * <p>
 * If the <tt>handleEvent</tt> method throws an unchecked runtime exception,
 * the JAXB Provider must treat that as if the method returned false, effectively
 * terminating whatever operation was in progress at the time (unmarshal,
 * validate, or marshal).
 *
 * <p>
 * Modifying the Java content tree within your event handler is undefined
 * by the specification and may result in unexpected behaviour.
 *
 * <p>
 * Failing to return false from the <tt>handleEvent</tt> method after
 * encountering a fatal error is undefined by the specification and may result
 * in unexpected behavior.
 *
 * <p>
 * <b>Default Event Handler</b>
 * <blockquote>
 *    See: <a href="Validator.html#defaulthandler">Validator javadocs</a>
```

```
 * </blockquote>
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see Unmarshaller
 * @see Validator
 * @see Marshaller
 * @see ValidationEvent
 * @see javax.xml.bind.util.ValidationEventCollector
 * @since JAXB1.0
 */
public interface ValidationEventHandler {
    /**
     * Receive notification of a validation warning or error.
     *
     * The ValidationEvent will have a
     * {@link ValidationEventLocator ValidationEventLocator} embedded in it that
     * indicates where the error or warning occurred.
     *
     * <p>
     * If an unchecked runtime exception is thrown from this method, the JAXB
     * provider will treat it as if the method returned false and interrupt
     * the current unmarshal, validate, or marshal operation.
     *
     * @param event the encapsulated validation event information.  It is a
     * provider error if this parameter is null.
     * @return true if the JAXB Provider should attempt to continue the current
     *         unmarshal, validate, or marshal operation after handling this
     *         warning/error, false if the provider should terminate the current
     *         operation with the appropriate <tt>UnmarshalException</tt>,
     *         <tt>ValidationException</tt>, or <tt>MarshalException</tt>.
     * @throws IllegalArgumentException if the event object is null.
     */
    public boolean handleEvent( ValidationEvent event );

}
```

2.29 ValidationEventLocator.java

Secuencia 31: javax/xml/bind/ValidationEventLocator.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * Encapsulate the location of a ValidationEvent.
 *
 * <p>
 * The <tt>ValidationEventLocator</tt> indicates where the <tt>ValidationEvent
 * </tt> occurred.  Different fields will be set depending on the type of
 * validation that was being performed when the error or warning was detected.
 * For example, on-demand validation would produce locators that contained
 * references to objects in the Java content tree while unmarshal-time
 * validation would produce locators containing information appropriate to the
 * source of the XML data (file, url, Node, etc).
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see Validator
 * @see ValidationEvent
 * @since JAXB1.0
 */
public interface ValidationEventLocator {

    /**
     * Return the name of the XML source as a URL if available
     *
     * @return the name of the XML source as a URL or null if unavailable
     */
    public java.net.URL getURL();

    /**
     * Return the byte offset if available
     *
     * @return the byte offset into the input source or -1 if unavailable
     */
    public int getOffset();

    /**
     * Return the line number if available
     *
     * @return the line number or -1 if unavailable
     */
```

```
    public int getLineNumber();

    /**
     * Return the column number if available
     *
     * @return the column number or -1 if unavailable
     */
    public int getColumnNumber();

    /**
     * Return a reference to the object in the Java content tree if available
     *
     * @return a reference to the object in the Java content tree or null if
     *         unavailable
     */
    public java.lang.Object getObject();

    /**
     * Return a reference to the DOM Node if available
     *
     * @return a reference to the DOM Node or null if unavailable
     */
    public org.w3c.dom.Node getNode();

}
```

2.30 ValidationException.java

Secuencia 32: javax/xml/bind/ValidationException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```java

package javax.xml.bind;

/**
 * This exception indicates that an error has occurred while performing
 * a validate operation.
 *
 * <p>
 * The <tt>ValidationEventHandler</tt> can cause this exception to be thrown
 * during the validate operations.  See
 * {@link ValidationEventHandler#handleEvent(ValidationEvent)
 * ValidationEventHandler.handleEvent(ValidationEvent)}.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see JAXBException
 * @see Validator
 * @since JAXB1.0
 */
public class ValidationException extends JAXBException {

    /**
     * Construct an ValidationException with the specified detail message.  The
     * errorCode and linkedException will default to null.
     *
     * @param message a description of the exception
     */
    public ValidationException(String message) {
        this( message, null, null );
    }

    /**
     * Construct an ValidationException with the specified detail message and vendor
     * specific errorCode.  The linkedException will default to null.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     */
    public ValidationException(String message, String errorCode) {
        this( message, errorCode, null );
    }

    /**
     * Construct an ValidationException with a linkedException.  The detail message and
     * vendor specific errorCode will default to null.
     *
     * @param exception the linked exception
     */
    public ValidationException(Throwable exception) {
        this( null, null, exception );
    }

    /**
     * Construct an ValidationException with the specified detail message and
```

```
     * linkedException.  The errorCode will default to null.
     *
     * @param message a description of the exception
     * @param exception the linked exception
     */
    public ValidationException(String message, Throwable exception) {
        this( message, null, exception );
    }

    /**
     * Construct an ValidationException with the specified detail message, vendor
     * specific errorCode, and linkedException.
     *
     * @param message a description of the exception
     * @param errorCode a string specifying the vendor specific error code
     * @param exception the linked exception
     */
    public ValidationException(String message, String errorCode, Throwable exception) {
        super( message, errorCode, exception );
    }

}
```

2.31 Validator.java

Secuencia 33: javax/xml/bind/Validator.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

```

```
/**
 * As of JAXB 2.0, this class is deprecated and optional.
 * <p>
 * The <tt>Validator</tt> class is responsible for controlling the validation
 * of content trees during runtime.
 *
 * <p>
 * <a name="validationtypes"></a>
 * <b>Three Forms of Validation</b><br>
 * <blockquote>
 *    <dl>
 *        <dt><b>Unmarshal-Time Validation</b></dt>
 *        <dd>This form of validation enables a client application to receive
 *            information about validation errors and warnings detected while
 *            unmarshalling XML data into a Java content tree and is completely
 *            orthogonal to the other types of validation.  To enable or disable
 *            it, see the javadoc for
 *            {@link Unmarshaller#setValidating(boolean) Unmarshaller.setValidating}.
 *            All JAXB 1.0 Providers are required to support this operation.
 *        </dd>
 *
 *        <dt><b>On-Demand Validation</b></dt>
 *        <dd> This form of validation enables a client application to receive
 *             information about validation errors and warnings detected in the
 *             Java content tree.  At any point, client applications can call
 *             the {@link Validator#validate(Object) Validator.validate} method
 *             on the Java content tree (or any sub-tree of it).  All JAXB 1.0
 *             Providers are required to support this operation.
 *        </dd>
 *
 *        <dt><b>Fail-Fast Validation</b></dt>
 *        <dd> This form of validation enables a client application to receive
 *             immediate feedback about modifications to the Java content tree
 *             that violate type constraints on Java Properties as defined in
 *             the specification.  JAXB Providers are not required support
 *             this type of validation.  Of the JAXB Providers that do support
 *             this type of validation, some may require you to decide at schema
 *             compile time whether or not a client application will be allowed
 *             to request fail-fast validation at runtime.
 *        </dd>
 *    </dl>
 * </blockquote>
 *
 * <p>
 * The <tt>Validator</tt> class is responsible for managing On-Demand Validation.
 * The <tt>Unmarshaller</tt> class is responsible for managing Unmarshal-Time
 * Validation during the unmarshal operations.  Although there is no formal
 * method of enabling validation during the marshal operations, the
 * <tt>Marshaller</tt> may detect errors, which will be reported to the
 * <tt>ValidationEventHandler</tt> registered on it.
 *
 * <p>
 * <a name="defaulthandler"></a>
```

```
* <b>Using the Default EventHandler</b><br>
* <blockquote>
*   If the client application does not set an event handler on their
*   <tt>Validator</tt>, <tt>Unmarshaller</tt>, or <tt>Marshaller</tt> prior to
*   calling the validate, unmarshal, or marshal methods, then a default event
*   handler will receive notification of any errors or warnings encountered.
*   The default event handler will cause the current operation to halt after
*   encountering the first error or fatal error (but will attempt to continue
*   after receiving warnings).
* </blockquote>
*
* <p>
* <a name="handlingevents"></a>
* <b>Handling Validation Events</b><br>
* <blockquote>
*   There are three ways to handle events encountered during the unmarshal,
*   validate, and marshal operations:
*    <dl>
*        <dt>Use the default event handler</dt>
*        <dd>The default event handler will be used if you do not specify one
*            via the <tt>setEventHandler</tt> API's on <tt>Validator</tt>,
*            <tt>Unmarshaller</tt>, or <tt>Marshaller</tt>.
*        </dd>
*
*        <dt>Implement and register a custom event handler</dt>
*        <dd>Client applications that require sophisticated event processing
*            can implement the <tt>ValidationEventHandler</tt> interface and
*            register it with the <tt>Unmarshaller</tt> and/or
*            <tt>Validator</tt>.
*        </dd>
*
*        <dt>Use the {@link javax.xml.bind.util.ValidationEventCollector ValidationEventCollector}
*            utility</dt>
*        <dd>For convenience, a specialized event handler is provided that
*            simply collects any <tt>ValidationEvent</tt> objects created
*            during the unmarshal, validate, and marshal operations and
*            returns them to the client application as a
*            <tt>java.util.Collection</tt>.
*        </dd>
*    </dl>
* </blockquote>
*
* <p>
* <b>Validation and Well-Formedness</b><br>
* <blockquote>
* <p>
* Validation events are handled differently depending on how the client
* application is configured to process them as described in the previous
* section.  However, there are certain cases where a JAXB Provider indicates
* that it is no longer able to reliably detect and report errors.  In these
* cases, the JAXB Provider will set the severity of the ValidationEvent to
* FATAL_ERROR to indicate that the unmarshal, validate, or marshal operations
* should be terminated.  The default event handler and
```

```
 * <tt>ValidationEventCollector</tt> utility class must terminate processing
 * after being notified of a fatal error.  Client applications that supply their
 * own <tt>ValidationEventHandler</tt> should also terminate processing after
 * being notified of a fatal error.  If not, unexpected behaviour may occur.
 * </blockquote>
 *
 * <p>
 * <a name="supportedProps"></a>
 * <b>Supported Properties</b><br>
 * <blockquote>
 * <p>
 * There currently are not any properties required to be supported by all
 * JAXB Providers on Validator.  However, some providers may support
 * their own set of provider specific properties.
 * </blockquote>
 *
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li><li>Kohsuke Kawaguchi, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see JAXBContext
 * @see Unmarshaller
 * @see ValidationEventHandler
 * @see ValidationEvent
 * @see javax.xml.bind.util.ValidationEventCollector
 * @since JAXB1.0
 * @deprecated since JAXB 2.0
 */
public interface Validator {

    /**
     * Allow an application to register a validation event handler.
     * <p>
     * The validation event handler will be called by the JAXB Provider if any
     * validation errors are encountered during calls to
     * {@link #validate(Object) validate}.  If the client application does not
     * register a validation event handler before invoking the validate method,
     * then validation events will be handled by the default event handler which
     * will terminate the validate operation after the first error or fatal error
     * is encountered.
     * <p>
     * Calling this method with a null parameter will cause the Validator
     * to revert back to the default default event handler.
     *
     * @param handler the validation event handler
     * @throws JAXBException if an error was encountered while setting the
     *         event handler
     * @deprecated since JAXB2.0
     */
    public void setEventHandler( ValidationEventHandler handler )
        throws JAXBException;

    /**
     * Return the current event handler or the default event handler if one
```

```
     * hasn't been set.
     *
     * @return the current ValidationEventHandler or the default event handler
     *         if it hasn't been set
     * @throws JAXBException if an error was encountered while getting the
     *         current event handler
     * @deprecated since JAXB2.0
     */
    public ValidationEventHandler getEventHandler()
        throws JAXBException;

    /**
     * Validate the Java content tree starting at <tt>subrootObj</tt>.
     * <p>
     * Client applications can use this method to validate Java content trees
     * on-demand at runtime.  This method can be used to validate any arbitrary
     * subtree of the Java content tree.  Global constraint checking <b>will not
     * </b> be performed as part of this operation (i.e. ID/IDREF constraints).
     *
     * @param subrootObj the obj to begin validation at
     * @throws JAXBException if any unexpected problem occurs during validation
     * @throws ValidationException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Validator</tt> is unable to validate the content tree rooted
     *     at <tt>subrootObj</tt>
     * @throws IllegalArgumentException
     *      If the subrootObj parameter is null
     * @return true if the subtree rooted at <tt>subrootObj</tt> is valid, false
     *         otherwise
     * @deprecated since JAXB2.0
     */
    public boolean validate( Object subrootObj ) throws JAXBException;

    /**
     * Validate the Java content tree rooted at <tt>rootObj</tt>.
     * <p>
     * Client applications can use this method to validate Java content trees
     * on-demand at runtime.  This method is used to validate an entire Java
     * content tree.  Global constraint checking <b>will</b> be performed as
     * part of this operation (i.e. ID/IDREF constraints).
     *
     * @param rootObj the root obj to begin validation at
     * @throws JAXBException if any unexpected problem occurs during validation
     * @throws ValidationException
     *     If the {@link ValidationEventHandler ValidationEventHandler}
     *     returns false from its <tt>handleEvent</tt> method or the
     *     <tt>Validator</tt> is unable to validate the content tree rooted
     *     at <tt>rootObj</tt>
     * @throws IllegalArgumentException
     *      If the rootObj parameter is null
     * @return true if the tree rooted at <tt>rootObj</tt> is valid, false
     *         otherwise
```

```
     * @deprecated since JAXB2.0
     */
    public boolean validateRoot( Object rootObj ) throws JAXBException;

    /**
     * Set the particular property in the underlying implementation of
     * <tt>Validator</tt>.  This method can only be used to set one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to set an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to be set. This value can either
     *              be specified using one of the constant fields or a user
     *              supplied string.
     * @param value the value of the property to be set
     *
     * @throws PropertyException when there is an error processing the given
     *                           property or value
     * @throws IllegalArgumentException
     *      If the name parameter is null
     * @deprecated since JAXB2.0
     */
    public void setProperty( String name, Object value )
        throws PropertyException;

    /**
     * Get the particular property in the underlying implementation of
     * <tt>Validator</tt>.  This method can only be used to get one of
     * the standard JAXB defined properties above or a provider specific
     * property.  Attempting to get an undefined property will result in
     * a PropertyException being thrown.  See <a href="#supportedProps">
     * Supported Properties</a>.
     *
     * @param name the name of the property to retrieve
     * @return the value of the requested property
     *
     * @throws PropertyException
     *      when there is an error retrieving the given property or value
     *      property name
     * @throws IllegalArgumentException
     *      If the name parameter is null
     * @deprecated since JAXB2.0
     */
    public Object getProperty( String name ) throws PropertyException;

}
```

2.32 WhiteSpaceProcessor.java

Secuencia 34: javax/xml/bind/WhiteSpaceProcessor.java

```
/*
 * Copyright (c) 2007, 2013, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind;

/**
 * Processes white space normalization.
 *
 * @since 1.0
 */
abstract class WhiteSpaceProcessor {

// benchmarking (see test/src/ReplaceTest.java in the CVS Attic)
// showed that this code is slower than the current code.
//
//    public static String replace(String text) {
//        final int len = text.length();
//        StringBuffer result = new StringBuffer(len);
//
//        for (int i = 0; i < len; i++) {
//            char ch = text.charAt(i);
//            if (isWhiteSpace(ch))
//                result.append(' ');
//            else
//                result.append(ch);
//        }
//
//        return result.toString();
//    }

    public static String replace(String text) {
        return replace( (CharSequence)text ).toString();
    }
```

```java

    /**
     * @since 2.0
     */
    public static CharSequence replace(CharSequence text) {
        int i=text.length()-1;

        // look for the first whitespace char.
        while( i>=0 && !isWhiteSpaceExceptSpace(text.charAt(i)) )
            i--;

        if( i<0 )
            // no such whitespace. replace(text)==text.
            return text;

        // we now know that we need to modify the text.
        // allocate a char array to do it.
        StringBuilder buf = new StringBuilder(text);

        buf.setCharAt(i--,' ');
        for( ; i>=0; i-- )
            if( isWhiteSpaceExceptSpace(buf.charAt(i)))
                buf.setCharAt(i,' ');

        return new String(buf);
    }

    /**
     * Equivalent of {@link String#trim()}.
     * @since 2.0
     */
    public static CharSequence trim(CharSequence text) {
        int len = text.length();
        int start = 0;

        while( start<len && isWhiteSpace(text.charAt(start)) )
            start++;

        int end = len-1;

        while( end>start && isWhiteSpace(text.charAt(end)) )
            end--;

        if(start==0 && end==len-1)
            return text;    // no change
        else
            return text.subSequence(start,end+1);
    }

    public static String collapse(String text) {
        return collapse( (CharSequence)text ).toString();
    }

```

```java
    /**
     * This is usually the biggest processing bottleneck.
     *
     * @since 2.0
     */
    public static CharSequence collapse(CharSequence text) {
        int len = text.length();

        // most of the texts are already in the collapsed form.
        // so look for the first whitespace in the hope that we will
        // never see it.
        int s=0;
        while(s<len) {
            if(isWhiteSpace(text.charAt(s)))
                break;
            s++;
        }
        if(s==len)
            // the input happens to be already collapsed.
            return text;

        // we now know that the input contains spaces.
        // let's sit down and do the collapsing normally.

        StringBuilder result = new StringBuilder(len /*allocate enough size to avoid re-allocation
            */ );

        if(s!=0) {
            for( int i=0; i<s; i++ )
                result.append(text.charAt(i));
            result.append(' ');
        }

        boolean inStripMode = true;
        for (int i = s+1; i < len; i++) {
            char ch = text.charAt(i);
            boolean b = isWhiteSpace(ch);
            if (inStripMode && b)
                continue; // skip this character

            inStripMode = b;
            if (inStripMode)
                result.append(' ');
            else
                result.append(ch);
        }

        // remove trailing whitespaces
        len = result.length();
        if (len > 0 && result.charAt(len - 1) == ' ')
            result.setLength(len - 1);
        // whitespaces are already collapsed,
        // so all we have to do is to remove the last one character
```

```
        // if it's a whitespace.

        return result;
    }

    /**
     * Returns true if the specified string is all whitespace.
     */
    public static final boolean isWhiteSpace(CharSequence s) {
        for( int i=s.length()-1; i>=0; i-- )
            if(!isWhiteSpace(s.charAt(i)))
                return false;
        return true;
    }

    /** returns true if the specified char is a white space character. */
    public static final boolean isWhiteSpace(char ch) {
        // most of the characters are non-control characters.
        // so check that first to quickly return false for most of the cases.
        if( ch>0x20 )   return false;

        // other than we have to do four comparisons.
        return ch == 0x9 || ch == 0xA || ch == 0xD || ch == 0x20;
    }

    /**
     * Returns true if the specified char is a white space character
     * but not 0x20.
     */
    protected static final boolean isWhiteSpaceExceptSpace(char ch) {
        // most of the characters are non-control characters.
        // so check that first to quickly return false for most of the cases.
        if( ch>=0x20 )   return false;

        // other than we have to do four comparisons.
        return ch == 0x9 || ch == 0xA || ch == 0xD;
    }
}
```

3 Xml Bind Annotation (javax.xml.bind.annotation package)

3.1 DomHandler.java

Secuencia 35: javax/xml/bind/annotation/DomHandler.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.ValidationEventHandler;
import javax.xml.transform.Result;
import javax.xml.transform.Source;

/**
 * Converts an element (and its descendants)
 * from/to DOM (or similar) representation.
 *
 * <p>
 * Implementations of this interface will be used in conjunction with
 * {@link XmlAnyElement} annotation to map an element of XML into a representation
 * of infoset such as W3C DOM.
 *
 * <p>
 * Implementations hide how a portion of XML is converted into/from such
 * DOM-like representation, allowing JAXB providers to work with arbitrary
 * such library.
 *
 * <P>
 * This interface is intended to be implemented by library writers
 * and consumed by JAXB providers. None of those methods are intended to
 * be called from applications.
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
public interface DomHandler<ElementT,ResultT extends Result> {
    /**
     * When a JAXB provider needs to unmarshal a part of a document into an
     * infoset representation, it first calls this method to create a
     * {@link Result} object.
     *
     * <p>
     * A JAXB provider will then send a portion of the XML
     * into the given result. Such a portion always form a subtree
```

```
     * of the whole XML document rooted at an element.
     *
     * @param errorHandler
     *      if any error happens between the invocation of this method
     *      and the invocation of {@link #getElement(Result)}, they
     *      must be reported to this handler.
     *
     *      The caller must provide a non-null error handler.
     *
     *      The {@link Result} object created from this method
     *      may hold a reference to this error handler.
     *
     * @return
     *      null if the operation fails. The error must have been reported
     *      to the error handler.
     */
    ResultT createUnmarshaller( ValidationEventHandler errorHandler );

    /**
     * Once the portion is sent to the {@link Result}. This method is called
     * by a JAXB provider to obtain the unmarshalled element representation.
     *
     * <p>
     * Multiple invocations of this method may return different objects.
     * This method can be invoked only when the whole sub-tree are fed
     * to the {@link Result} object.
     *
     * @param rt
     *      The {@link Result} object created by {@link #createUnmarshaller(ValidationEventHandler)
         }.
     *
     * @return
     *      null if the operation fails. The error must have been reported
     *      to the error handler.
     */
    ElementT getElement(ResultT rt);

    /**
     * This method is called when a JAXB provider needs to marshal an element
     * to XML.
     *
     * <p>
     * If non-null, the returned {@link Source} must contain a whole document
     * rooted at one element, which will then be weaved into a bigger document
     * that the JAXB provider is marshalling.
     *
     * @param errorHandler
     *      Receives any errors happened during the process of converting
     *      an element into a {@link Source}.
     *
     *      The caller must provide a non-null error handler.
     *
     * @return
```

```
     *      null if there was an error. The error should have been reported
     *      to the handler.
     */
    Source marshal( ElementT n, ValidationEventHandler errorHandler );
}
```

3.2 W3CDomHandler.java

Secuencia 36: javax/xml/bind/annotation/W3CDomHandler.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import org.w3c.dom.Document;
import org.w3c.dom.DocumentFragment;
import org.w3c.dom.Element;
import org.w3c.dom.Node;

import javax.xml.bind.ValidationEventHandler;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.transform.Source;
import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.dom.DOMSource;

/**
 * {@link DomHandler} implementation for W3C DOM (<code>org.w3c.dom</code> package.)
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
```

```java
public class W3CDomHandler implements DomHandler<Element,DOMResult> {

    private DocumentBuilder builder;

    /**
     * Default constructor.
     *
     * It is up to a JAXB provider to decide which DOM implementation
     * to use or how that is configured.
     */
    public W3CDomHandler() {
        this.builder = null;
    }

    /**
     * Constructor that allows applications to specify which DOM implementation
     * to be used.
     *
     * @param builder
     *      must not be null. JAXB uses this {@link DocumentBuilder} to create
     *      a new element.
     */
    public W3CDomHandler(DocumentBuilder builder) {
        if(builder==null)
            throw new IllegalArgumentException();
        this.builder = builder;
    }

    public DocumentBuilder getBuilder() {
        return builder;
    }

    public void setBuilder(DocumentBuilder builder) {
        this.builder = builder;
    }

    public DOMResult createUnmarshaller(ValidationEventHandler errorHandler) {
        if(builder==null)
            return new DOMResult();
        else
            return new DOMResult(builder.newDocument());
    }

    public Element getElement(DOMResult r) {
        // JAXP spec is ambiguous about what really happens in this case,
        // so work defensively
        Node n = r.getNode();
        if( n instanceof Document ) {
            return ((Document)n).getDocumentElement();
        }
        if( n instanceof Element )
            return (Element)n;
        if( n instanceof DocumentFragment )
```

```
            return (Element)n.getChildNodes().item(0);

        // if the result object contains something strange,
        // it is not a user problem, but it is a JAXB provider's problem.
        // That's why we throw a runtime exception.
        throw new IllegalStateException(n.toString());
    }

    public Source marshal(Element element, ValidationEventHandler errorHandler) {
        return new DOMSource(element);
    }
}
```

3.3 XmlAccessOrder.java

Secuencia 37: javax/xml/bind/annotation/XmlAccessOrder.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

/**
 * Used by XmlAccessorOrder to control the ordering of properties and
 * fields in a JAXB bound class.
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 * @see XmlAccessorOrder
 */

public enum XmlAccessOrder {
```

```
    /**
     * The ordering of fields and properties in a class is undefined.
     */
    UNDEFINED,
    /**
     * The ordering of fields and properties in a class is in
     * alphabetical order as determined by the
     * method java.lang.String.compareTo(String anotherString).
     */
    ALPHABETICAL
}
```

3.4 XmlAccessType.java

Secuencia 38: javax/xml/bind/annotation/XmlAccessType.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;



/**
 * Used by XmlAccessorType to control serialization of fields or
 * properties.
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 * @see XmlAccessorType
 */

```

```
public enum XmlAccessType {
    /**
     * Every getter/setter pair in a JAXB-bound class will be automatically
     * bound to XML, unless annotated by {@link XmlTransient}.
     *
     * Fields are bound to XML only when they are explicitly annotated
     * by some of the JAXB annotations.
     */
    PROPERTY,
    /**
     * Every non static, non transient field in a JAXB-bound class will be automatically
     * bound to XML, unless annotated by {@link XmlTransient}.
     *
     * Getter/setter pairs are bound to XML only when they are explicitly annotated
     * by some of the JAXB annotations.
     */
    FIELD,
    /**
     * Every public getter/setter pair and every public field will be
     * automatically bound to XML, unless annotated by {@link XmlTransient}.
     *
     * Fields or getter/setter pairs that are private, protected, or
     * defaulted to package-only access are bound to XML only when they are
     * explicitly annotated by the appropriate JAXB annotations.
     */
    PUBLIC_MEMBER,
    /**
     * None of the fields or properties is bound to XML unless they
     * are specifically  annotated with some of the JAXB annotations.
     */
    NONE
}
```

3.5 XmlAccessorOrder.java

Secuencia 39: javax/xml/bind/annotation/XmlAccessorOrder.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.Inherited;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p> Controls the ordering of fields and properties in a class. </p>
 *
 * <h3>Usage </h3>
 *
 * <p> <tt> @XmlAccessorOrder </tt> annotation can be used with the following
 * program elements:</p>
 *
 * <ul>
 *   <li> package</li>
 *   <li> a top level class </li>
 * </ul>
 *
 * <p> See "Package Specification" in <tt>javax.xml.bind</tt> package javadoc for
 * additional common information.</p>
 *
 * <p>The effective {@link XmlAccessOrder} on a class is determined
 * as follows:
 *
 * <ul>
 *   <li> If there is a <tt>@XmlAccessorOrder</tt> on a class, then
 *        it is used. </li>
 *   <li> Otherwise, if a <tt>@XmlAccessorOrder </tt> exists on one of
 *        its super classes, then it is inherited (by the virtue of
 *        {@link Inherited})
 *   <li> Otherwise, the <tt>@XmlAccessorOrder</tt> on the package
 *        of the class is used, if it's there.
 *   <li> Otherwise {@link XmlAccessOrder#UNDEFINED}.
 * </ul>
 *
 * <p>This annotation can be used with the following annotations:
 *    {@link XmlType}, {@link XmlRootElement}, {@link XmlAccessorType},
 *    {@link XmlSchema}, {@link XmlSchemaType}, {@link XmlSchemaTypes},
 *    , {@link XmlJavaTypeAdapter}. It can also be used with the
 *    following annotations at the package level: {@link XmlJavaTypeAdapter}.
```

```
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 * @see XmlAccessOrder
 */

@Inherited @Retention(RUNTIME) @Target({PACKAGE, TYPE})
public @interface XmlAccessorOrder {
        XmlAccessOrder value() default XmlAccessOrder.UNDEFINED;
}
```

3.6 XmlAccessorType.java

Secuencia 40: javax/xml/bind/annotation/XmlAccessorType.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.Inherited;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p> Controls whether fields or Javabean properties are serialized by default. </p>
 *
 * <p> <b> Usage </b> </p>
```

```
 *
 * <p> <tt>@XmlAccessorType</tt> annotation can be used with the following program elements:</p>
 *
 * <ul>
 *   <li> package</li>
 *   <li> a top level class </li>
 * </ul>
 *
 * <p> See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p>This annotation provides control over the default serialization
 * of properties and fields in a class.
 *
 * <p>The annotation <tt> @XmlAccessorType </tt> on a package applies to
 * all classes in the package. The following inheritance
 * semantics apply:
 *
 * <ul>
 *   <li> If there is a <tt>@XmlAccessorType</tt> on a class, then it
 *        is used. </li>
 *   <li> Otherwise, if a <tt>@XmlAccessorType</tt> exists on one of
 *        its super classes, then it is inherited.
 *   <li> Otherwise, the <tt>@XmlAccessorType </tt> on a package is
 *        inherited.
 * </ul>
 * <p> <b> Defaulting Rules: </b> </p>
 *
 * <p>By default, if <tt>@XmlAccessorType </tt> on a package is absent,
 * then the following package level annotation is assumed.</p>
 * <pre>
 *   &#64;XmlAccessorType(XmlAccessType.PUBLIC_MEMBER)
 * </pre>
 * <p> By default, if <tt>@XmlAccessorType</tt> on a class is absent,
 * and none of its super classes is annotated with
 * <tt>@XmlAccessorType</tt>, then the following default on the class
 * is assumed: </p>
 * <pre>
 *   &#64;XmlAccessorType(XmlAccessType.PUBLIC_MEMBER)
 * </pre>
 * <p>This annotation can be used with the following annotations:
 *    {@link XmlType}, {@link XmlRootElement}, {@link XmlAccessorOrder},
 *    {@link XmlSchema}, {@link XmlSchemaType}, {@link XmlSchemaTypes},
 *    , {@link XmlJavaTypeAdapter}. It can also be used with the
 *    following annotations at the package level: {@link XmlJavaTypeAdapter}.
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 * @see XmlAccessType
 */

@Inherited @Retention(RUNTIME) @Target({PACKAGE, TYPE})
public @interface XmlAccessorType {
```

```

    /**
     * Specifies whether fields or properties are serialized.
     *
     * @see XmlAccessType
     */
    XmlAccessType value() default XmlAccessType.PUBLIC_MEMBER;
}
```

3.7 XmlAnyAttribute.java

Secuencia 41: javax/xml/bind/annotation/XmlAnyAttribute.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.namespace.QName;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import java.util.Map;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;

/**
 * <p>
 * Maps a JavaBean property to a map of wildcard attributes.
 *
 * <p> <b>Usage</b> </p>
```

```
 * <p>
 * The <tt>&#64;XmlAnyAttribute</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li> JavaBean property </li>
 *   <li> non static, non transient field </li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * The usage is subject to the following constraints:
 * <ul>
 *   <li> At most one field or property in a class can be annotated
 *        with <tt>&#64;XmlAnyAttribute</tt>.  </li>
 *   <li> The type of the property or the field must <tt>java.util.Map</tt> </li>
 * </ul>
 *
 * <p>
 * While processing attributes to be unmarshalled into a value class,
 * each attribute that is not statically associated with another
 * JavaBean property, via {@link XmlAttribute}, is entered into the
 * wildcard attribute map represented by
 * {@link Map}&lt;{@link QName},{@link Object}>. The attribute QName is the
 * map's key. The key's value is the String value of the attribute.
 *
 * @author Kohsuke Kawaguchi, Sun Microsystems, Inc.
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD})
public @interface XmlAnyAttribute {
}
```

3.8 XmlAnyElement.java

Secuencia 42: javax/xml/bind/annotation/XmlAnyElement.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import org.w3c.dom.Element;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import java.util.List;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * Maps a JavaBean property to XML infoset representation and/or JAXB element.
 *
 * <p>
 * This annotation serves as a "catch-all" property while unmarshalling
 * xml content into a instance of a JAXB annotated class. It typically
 * annotates a multi-valued JavaBean property, but it can occur on
 * single value JavaBean property. During unmarshalling, each xml element
 * that does not match a static &#64;XmlElement or &#64;XmlElementRef
 * annotation for the other JavaBean properties on the class, is added to this
 * "catch-all" property.
 *
 * <p>
 * <h2>Usages:</h2>
 * <pre>
 * &#64;XmlAnyElement
 * public {@link Element}[] others;
 *
 * // Collection of {@link Element} or JAXB elements.
 * &#64;XmlAnyElement(lax="true")
 * public {@link Object}[] others;
 *
 * &#64;XmlAnyElement
 * private List&lt;{@link Element}> nodes;
 *
 * &#64;XmlAnyElement
 * private {@link Element} node;
 * </pre>
 *
```

```
* <h2>Restriction usage constraints</h2>
* <p>
* This annotation is mutually exclusive with
* {@link XmlElement}, {@link XmlAttribute}, {@link XmlValue},
* {@link XmlElements}, {@link XmlID}, and {@link XmlIDREF}.
*
* <p>
* There can be only one {@link XmlAnyElement} annotated JavaBean property
* in a class and its super classes.
*
* <h2>Relationship to other annotations</h2>
* <p>
* This annotation can be used with {@link XmlJavaTypeAdapter}, so that users
* can map their own data structure to DOM, which in turn can be composed
* into XML.
*
* <p>
* This annotation can be used with {@link XmlMixed} like this:
* <pre>
* // List of java.lang.String or DOM nodes.
* &#64;XmlAnyElement &#64;XmlMixed
* List&lt;Object> others;
* </pre>
*
*
* <h2>Schema To Java example</h2>
*
* The following schema would produce the following Java class:
* <pre>
* &lt;xs:complexType name="foo">
*   &lt;xs:sequence>
*     &lt;xs:element name="a" type="xs:int" />
*     &lt;xs:element name="b" type="xs:int" />
*     &lt;xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
*   &lt;/xs:sequence>
* &lt;/xs:complexType>
* </pre>
*
* <pre>
* class Foo {
*   int a;
*   int b;
*   &#64;{@link XmlAnyElement}
*   List&lt;Element> any;
* }
* </pre>
*
* It can unmarshal instances like
*
* <pre>
* &lt;foo xmlns:e="extra">
*   &lt;a>1</a>
*   &lt;e:other />  // this will be bound to DOM, because unmarshalling is orderless
```

```
*   &lt;b>3</b>
*   &lt;e:other />
*   &lt;c>5</c>     // this will be bound to DOM, because the annotation doesn't remember namespaces.
* &lt;/foo>
* </pre>
*
*
*
* The following schema would produce the following Java class:
* <pre>
* &lt;xs:complexType name="bar">
*   &lt;xs:complexContent>
*   &lt;xs:extension base="foo">
*     &lt;xs:sequence>
*       &lt;xs:element name="c" type="xs:int" />
*       &lt;xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
*     &lt;/xs:sequence>
*   &lt;/xs:extension>
* &lt;/xs:complexType>
* </pre>
*
* <pre>
* class Bar extends Foo {
*   int c;
*   // Foo.getAny() also represents wildcard content for type definition bar.
* }
* </pre>
*
*
* It can unmarshal instances like
*
* <pre>
* &lt;bar xmlns:e="extra">
*   &lt;a>1</a>
*   &lt;e:other />  // this will be bound to DOM, because unmarshalling is orderless
*   &lt;b>3</b>
*   &lt;e:other />
*   &lt;c>5</c>     // this now goes to Bar.c
*   &lt;e:other />  // this will go to Foo.any
* &lt;/bar>
* </pre>
*
*
*
*
* <h2>Using {@link XmlAnyElement} with {@link XmlElementRef}</h2>
* <p>
* The {@link XmlAnyElement} annotation can be used with {@link XmlElementRef}s to
* designate additional elements that can participate in the content tree.
*
* <p>
```

```
* The following schema would produce the following Java class:
* <pre>
* &lt;xs:complexType name="foo">
*   &lt;xs:choice maxOccurs="unbounded" minOccurs="0">
*     &lt;xs:element name="a" type="xs:int" />
*     &lt;xs:element name="b" type="xs:int" />
*     &lt;xs:any namespace="##other" processContents="lax" />
*   &lt;/xs:choice>
* &lt;/xs:complexType>
* </pre>
*
* <pre>
* class Foo {
*   &#64;{@link XmlAnyElement}(lax="true")
*   &#64;{@link XmlElementRefs}({
*     &#64;{@link XmlElementRef}(name="a", type="JAXBElement.class")
*     &#64;{@link XmlElementRef}(name="b", type="JAXBElement.class")
*   })
*   {@link List}&lt;{@link Object}> others;
* }
*
* &#64;XmlRegistry
* class ObjectFactory {
*   ...
*   &#64;XmlElementDecl(name = "a", namespace = "", scope = Foo.class)
*   {@link JAXBElement}&lt;Integer> createFooA( Integer i ) { ... }
*
*   &#64;XmlElementDecl(name = "b", namespace = "", scope = Foo.class)
*   {@link JAXBElement}&lt;Integer> createFooB( Integer i ) { ... }
* </pre>
*
* It can unmarshal instances like
*
* <pre>
* &lt;foo xmlns:e="extra">
*   &lt;a>1</a>     // this will unmarshal to a {@link JAXBElement} instance whose value is 1.
*   &lt;e:other />  // this will unmarshal to a DOM {@link Element}.
*   &lt;b>3</b>     // this will unmarshal to a {@link JAXBElement} instance whose value is 1.
* &lt;/foo>
* </pre>
*
*
*
*
* <h2>W3C XML Schema "lax" wildcard emulation</h2>
* The lax element of the annotation enables the emulation of the "lax" wildcard semantics.
* For example, when the Java source code is annotated like this:
* <pre>
* &#64;{@link XmlRootElement}
* class Foo {
*   &#64;XmlAnyElement(lax=true)
*   public {@link Object}[] others;
* }
```

```
 * </pre>
 * then the following document will unmarshal like this:
 * <pre>
 * &lt;foo>
 *   &lt;unknown />
 *   &lt;foo />
 * &lt;/foo>
 *
 * Foo foo = unmarshal();
 * // 1 for 'unknown', another for 'foo'
 * assert foo.others.length==2;
 * // 'unknown' unmarshals to a DOM element
 * assert foo.others[0] instanceof Element;
 * // because of lax=true, the 'foo' element eagerly
 * // unmarshals to a Foo object.
 * assert foo.others[1] instanceof Foo;
 * </pre>
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD})
public @interface XmlAnyElement {

    /**
     * Controls the unmarshaller behavior when it sees elements
     * known to the current {@link JAXBContext}.
     *
     * <h3>When false</h3>
     * <p>
     * If false, all the elements that match the property will be unmarshalled
     * to DOM, and the property will only contain DOM elements.
     *
     * <h3>When true</h3>
     * <p>
     * If true, when an element matches a property marked with {@link XmlAnyElement}
     * is known to {@link JAXBContext} (for example, there's a class with
     * {@link XmlRootElement} that has the same tag name, or there's
     * {@link XmlElementDecl} that has the same tag name),
     * the unmarshaller will eagerly unmarshal this element to the JAXB object,
     * instead of unmarshalling it to DOM. Additionally, if the element is
     * unknown but it has a known xsi:type, the unmarshaller eagerly unmarshals
     * the element to a {@link JAXBElement}, with the unknown element name and
     * the JAXBElement value is set to an instance of the JAXB mapping of the
     * known xsi:type.
     *
     * <p>
     * As a result, after the unmarshalling, the property can become heterogeneous;
     * it can have both DOM nodes and some JAXB objects at the same time.
     *
     * <p>
     * This can be used to emulate the "lax" wildcard semantics of the W3C XML Schema.
```

```
     */
    boolean lax() default false;

    /**
     * Specifies the {@link DomHandler} which is responsible for actually
     * converting XML from/to a DOM-like data structure.
     */
    Class<? extends DomHandler> value() default W3CDomHandler.class;
}
```

3.9 XmlAttachmentRef.java

Secuencia 43: javax/xml/bind/annotation/XmlAttachmentRef.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.activation.DataHandler;
import static java.lang.annotation.ElementType.*;
import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * Marks a field/property that its XML form is a uri reference to mime content.
 * The mime content is optimally stored out-of-line as an attachment.
 *
 * A field/property must always map to the {@link DataHandler} class.
 *
 * <h2>Usage</h2>
```

```
 * <pre>
 * &#64;{@link XmlRootElement}
 * class Foo {
 *   &#64;{@link XmlAttachmentRef}
 *   &#64;{@link XmlAttribute}
 *   {@link DataHandler} data;
 *
 *   &#64;{@link XmlAttachmentRef}
 *   &#64;{@link XmlElement}
 *   {@link DataHandler} body;
 * }
 * </pre>
 * The above code maps to the following XML:
 * <pre>
 * &lt;xs:element name="foo" xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd">
 *   &lt;xs:complexType>
 *     &lt;xs:sequence>
 *       &lt;xs:element name="body" type="ref:swaRef" minOccurs="0" />
 *     &lt;/xs:sequence>
 *     &lt;xs:attribute name="data" type="ref:swaRef" use="optional" />
 *   &lt;/xs:complexType>
 * &lt;/xs:element>
 * </pre>
 *
 * <p>
 * The above binding supports WS-I AP 1.0 <a href="http://www.ws-i.org/Profiles/AttachmentsProfile
     -1.0-2004-08-24.html#Referencing_Attachments_from_the_SOAP_Envelope">WS-I Attachments Profile
     Version 1.0.</a>
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD,PARAMETER})
public @interface XmlAttachmentRef {
}
```

3.10 XmlAttribute.java

Secuencia 44: javax/xml/bind/annotation/XmlAttribute.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p>
 * Maps a JavaBean property to a XML attribute.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * The <tt>@XmlAttribute</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li> JavaBean property </li>
 *   <li> field </li>
 * </ul>
 *
 * <p> A static final field is mapped to a XML fixed attribute.
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * The usage is subject to the following constraints:
 * <ul>
 *   <li> If type of the field or the property is a collection
 *        type, then the collection item type must be mapped to schema
 *        simple type.
 * <pre>
 *     // Examples
 *     &#64;XmlAttribute List&lt;Integer> items; //legal
 *     &#64;XmlAttribute List&lt;Bar> foo; // illegal if Bar does not map to a schema simple type
 * </pre>
 *   </li>
 *   <li> If the type of the field or the property is a non
 *         collection type, then the type of the property or field
 *         must map to a simple schema type.
 * <pre>
```

```
*     // Examples
*     &#64;XmlAttribute int foo; // legal
*     &#64;XmlAttribute Foo foo; // illegal if Foo does not map to a schema simple type
* </pre>
*   </li>
*   <li> This annotation can be used with the following annotations:
*           {@link XmlID},
*           {@link XmlIDREF},
*           {@link XmlList},
*           {@link XmlSchemaType},
*           {@link XmlValue},
*           {@link XmlAttachmentRef},
*           {@link XmlMimeType},
*           {@link XmlInlineBinaryData},
*           {@link javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter}.</li>
* </ul>
* </p>
*
* <p> <b>Example 1: </b>Map a JavaBean property to an XML attribute.</p>
* <pre>
*     //Example: Code fragment
*     public class USPrice {
*         &#64;XmlAttribute
*         public java.math.BigDecimal getPrice() {...} ;
*         public void setPrice(java.math.BigDecimal ) {...};
*     }
*
*     &lt;!-- Example: XML Schema fragment -->
*     &lt;xs:complexType name="USPrice">
*       &lt;xs:sequence>
*       &lt;/xs:sequence>
*       &lt;xs:attribute name="price" type="xs:decimal"/>
*     &lt;/xs:complexType>
* </pre>
*
* <p> <b>Example 2: </b>Map a JavaBean property to an XML attribute with anonymous type.</p>
* See Example 7 in @{@link XmlType}.
*
* <p> <b>Example 3: </b>Map a JavaBean collection property to an XML attribute.</p>
* <pre>
*     // Example: Code fragment
*     class Foo {
*         ...
*         &#64;XmlAttribute List&lt;Integer> items;
*     }
*
*     &lt;!-- Example: XML Schema fragment -->
*     &lt;xs:complexType name="foo">
*       ...
*       &lt;xs:attribute name="items">
*         &lt;xs:simpleType>
*           &lt;xs:list itemType="xs:int"/>
*         &lt;/xs:simpleType>
```

```
 *     &lt;/xs:complexType>
 *
 * </pre>
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @see XmlType
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlAttribute {
    /**
     * Name of the XML Schema attribute. By default, the XML Schema
     * attribute name is derived from the JavaBean property name.
     *
     */
    String name() default "##default";

    /**
     * Specifies if the XML Schema attribute is optional or
     * required. If true, then the JavaBean property is mapped to a
     * XML Schema attribute that is required. Otherwise it is mapped
     * to a XML Schema attribute that is optional.
     *
     */
     boolean required() default false;

    /**
     * Specifies the XML target namespace of the XML Schema
     * attribute.
     *
     */
    String namespace() default "##default" ;
}
```

3.11 XmlElement.java

Secuencia 45: javax/xml/bind/annotation/XmlElement.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * Maps a JavaBean property to a XML element derived from property name.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * <tt>@XmlElement</tt> annotation can be used with the following program
 * elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> non static, non transient field </li>
 *   <li> within {@link XmlElements}
 * <p>
 *
 * </ul>
 *
 * The usage is subject to the following constraints:
 * <ul>
 *   <li> This annotation can be used with following annotations:
 *           {@link XmlID},
 *           {@link XmlIDREF},
 *           {@link XmlList},
 *           {@link XmlSchemaType},
 *           {@link XmlValue},
 *           {@link XmlAttachmentRef},
 *           {@link XmlMimeType},
 *           {@link XmlInlineBinaryData},
 *           {@link XmlElementWrapper},
 *           {@link XmlJavaTypeAdapter}</li>
 *   <li> if the type of JavaBean property is a collection type of
 *        array, an indexed property, or a parameterized list, and
 *        this annotation is used with {@link XmlElements} then,
 *        <tt>@XmlElement.type()</tt> must be DEFAULT.class since the
 *        collection item type is already known. </li>
 * </ul>
```

```
*
* <p>
* A JavaBean property, when annotated with @XmlElement annotation
* is mapped to a local element in the XML Schema complex type to
* which the containing class is mapped.
*
* <p>
* <b>Example 1: </b> Map a public non static non final field to local
* element
* <pre>
*     //Example: Code fragment
*     public class USPrice {
*         &#64;XmlElement(name="itemprice")
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example: Local XML Schema element -->
*     &lt;xs:complexType name="USPrice"/>
*       &lt;xs:sequence>
*         &lt;xs:element name="itemprice" type="xs:decimal" minOccurs="0"/>
*       &lt;/sequence>
*     &lt;/xs:complexType>
*   </pre>
* <p>
*
* <b> Example 2: </b> Map a field to a nillable element.
*   <pre>
*
*     //Example: Code fragment
*     public class USPrice {
*         &#64;XmlElement(nillable=true)
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example: Local XML Schema element -->
*     &lt;xs:complexType name="USPrice">
*       &lt;xs:sequence>
*         &lt;xs:element name="price" type="xs:decimal" nillable="true" minOccurs="0"/>
*       &lt;/sequence>
*     &lt;/xs:complexType>
*   </pre>
* <p>
* <b> Example 3: </b> Map a field to a nillable, required element.
*   <pre>
*
*     //Example: Code fragment
*     public class USPrice {
*         &#64;XmlElement(nillable=true, required=true)
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example: Local XML Schema element -->
*     &lt;xs:complexType name="USPrice">
```

```
 *       &lt;xs:sequence>
 *         &lt;xs:element name="price" type="xs:decimal" nillable="true" minOccurs="1"/>
 *       &lt;/sequence>
 *     &lt;/xs:complexType>
 *   </pre>
 * <p>
 *
 * <p> <b>Example 4: </b>Map a JavaBean property to an XML element
 * with anonymous type.</p>
 * <p>
 * See Example 6 in @{@link XmlType}.
 *
 * <p>
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD, PARAMETER})
public @interface XmlElement {
    /**
     * Name of the XML Schema element.
     * <p> If the value is "##default", then element name is derived from the
     * JavaBean property name.
     */
    String name() default "##default";

    /**
     * Customize the element declaration to be nillable.
     * <p>If nillable() is true, then the JavaBean property is
     * mapped to a XML Schema nillable element declaration.
     */
    boolean nillable() default false;

    /**
     * Customize the element declaration to be required.
     * <p>If required() is true, then Javabean property is mapped to
     * an XML schema element declaration with minOccurs="1".
     * maxOccurs is "1" for a single valued property and "unbounded"
     * for a multivalued property.
     * <p>If required() is false, then the Javabean property is mapped
     * to XML Schema element declaration with minOccurs="0".
     * maxOccurs is "1" for a single valued property and "unbounded"
     * for a multivalued property.
     */

    boolean required() default false;

    /**
     * XML target namespace of the XML Schema element.
     * <p>
     * If the value is "##default", then the namespace is determined
     * as follows:
     * <ol>
```

```
     *  <li>
     *  If the enclosing package has {@link XmlSchema} annotation,
     *  and its {@link XmlSchema#elementFormDefault() elementFormDefault}
     *  is {@link XmlNsForm#QUALIFIED QUALIFIED}, then the namespace of
     *  the enclosing class.
     *
     *  <li>
     *  Otherwise &#39;&#39; (which produces unqualified element in the default
     *  namespace.
     * </ol>
     */
    String namespace() default "##default";

    /**
     * Default value of this element.
     *
     * <p>
     * The <pre>'\u0000'</pre> value specified as a default of this annotation element
     * is used as a poor-man's substitute for null to allow implementations
     * to recognize the 'no default value' state.
     */
    String defaultValue() default "\u0000";

    /**
     * The Java class being referenced.
     */
    Class type() default DEFAULT.class;

    /**
     * Used in {@link XmlElement#type()} to
     * signal that the type be inferred from the signature
     * of the property.
     */
    static final class DEFAULT {}
}
```

3.12 XmlElementDecl.java

Secuencia 46: javax/xml/bind/annotation/XmlElementDecl.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.METHOD;

/**
 * Maps a factory method to a XML element.
 *
 * <p> <b>Usage</b> </p>
 *
 * The annotation creates a mapping between an XML schema element
 * declaration and a <i> element factory method </i> that returns a
 * JAXBElement instance representing the element
 * declaration. Typically, the element factory method is generated
 * (and annotated) from a schema into the ObjectFactory class in a
 * Java package that represents the binding of the element
 * declaration's target namespace. Thus, while the annotation syntax
 * allows &#64;XmlElementDecl to be used on any method, semantically
 * its use is restricted to annotation of element factory method.
 *
 * The usage is subject to the following constraints:
 *
 * <ul>
 *   <li> The class containing the element factory method annotated
 *        with &#64;XmlElementDecl must be marked with {@link
 *        XmlRegistry}. </li>
 *   <li> The element factory method must take one parameter
 *        assignable to {@link Object}.</li>
 * </ul>
 *
 * <p><b>Example 1: </b>Annotation on a factory method
 * <pre>
 *     // Example: code fragment
 *     &#64;XmlRegistry
 *     class ObjectFactory {
 *         &#64;XmlElementDecl(name="foo")
 *         JAXBElement&lt;String> createFoo(String s) { ... }
 *     }
 * </pre>
```

```
* <pre>
*     &lt;!-- XML input -->
*       &lt;foo>string</foo>
*
*     // Example: code fragment corresponding to XML input
*     JAXBElement<String> o =
*     (JAXBElement<String>)unmarshaller.unmarshal(aboveDocument);
*     // print JAXBElement instance to show values
*     System.out.println(o.getName());   // prints  "{}foo"
*     System.out.println(o.getValue());  // prints  "string"
*     System.out.println(o.getValue().getClass()); // prints "java.lang.String"
*
*     &lt;!-- Example: XML schema definition -->
*     &lt;xs:element name="foo" type="xs:string"/>
* </pre>
*
* <p><b>Example 2: </b> Element declaration with non local scope
* <p>
* The following example illustrates the use of scope annotation
* parameter in binding of element declaration in schema derived
* code.
* <p>
* The following example may be replaced in a future revision of
* this javadoc.
*
* <pre>
*     &lt;!-- Example: XML schema definition -->
*     &lt;xs:schema>
*       &lt;xs:complexType name="pea">
*         &lt;xs:choice maxOccurs="unbounded">
*           &lt;xs:element name="foo" type="xs:string"/>
*           &lt;xs:element name="bar" type="xs:string"/>
*         &lt;/xs:choice>
*       &lt;/xs:complexType>
*       &lt;xs:element name="foo" type="xs:int"/>
*     &lt;/xs:schema>
* </pre>
* <pre>
*     // Example: expected default binding
*     class Pea {
*         &#64;XmlElementRefs({
*             &#64;XmlElementRef(name="foo",type=JAXBElement.class)
*             &#64;XmlElementRef(name="bar",type=JAXBElement.class)
*         })
*         List&lt;JAXBElement&lt;String>> fooOrBar;
*     }
*
*     &#64;XmlRegistry
*     class ObjectFactory {
*         &#64;XmlElementDecl(scope=Pea.class,name="foo")
*         JAXBElement<String> createPeaFoo(String s);
*
*         &#64;XmlElementDecl(scope=Pea.class,name="bar")
```

```
 *         JAXBElement<String> createPeaBar(String s);
 *
 *         &#64;XmlElementDecl(name="foo")
 *         JAXBElement<Integer> createFoo(Integer i);
 *     }
 *
 * </pre>
 * Without scope createFoo and createPeaFoo would become ambiguous
 * since both of them map to a XML schema element with the same local
 * name "foo".
 *
 * @see XmlRegistry
 * @since JAXB 2.0
 */
@Retention(RUNTIME)
@Target({METHOD})
public @interface XmlElementDecl {
    /**
     * scope of the mapping.
     *
     * <p>
     * If this is not {@link XmlElementDecl.GLOBAL}, then this element
     * declaration mapping is only active within the specified class.
     */
    Class scope() default GLOBAL.class;

    /**
     * namespace name of the XML element.
     * <p>
     * If the value is "##default", then the value is the namespace
     * name for the package of the class containing this factory method.
     *
     * @see #name()
     */
    String namespace() default "##default";

    /**
     * local name of the XML element.
     *
     * <p>
     * <b> Note to reviewers: </b> There is no default name; since
     * the annotation is on a factory method, it is not clear that the
     * method name can be derived from the factory method name.
     * @see #namespace()
     */
    String name();

    /**
     * namespace name of a substitution group's head XML element.
     * <p>
     * This specifies the namespace name of the XML element whose local
     * name is specified by <tt>substitutionHeadName()</tt>.
     * <p>
```

```
     * If <tt>susbtitutionHeadName()</tt> is "", then this
     * value can only be "##default". But the value is ignored since
     * since this element is not part of susbtitution group when the
     * value of <tt>susbstitutionHeadName()</tt> is "".
     * <p>
     * If <tt>susbtitutionHeadName()</tt> is not "" and the value is
     * "##default", then the namespace name is the namespace name to
     * which the package of the containing class, marked with {@link
     * XmlRegistry }, is mapped.
     * <p>
     * If <tt>susbtitutionHeadName()</tt> is not "" and the value is
     * not "##default", then the value is the namespace name.
     *
     * @see #substitutionHeadName()
     */
    String substitutionHeadNamespace() default "##default";

    /**
     * XML local name of a substitution group's head element.
     * <p>
     * If the value is "", then this element is not part of any
     * substitution group.
     *
     * @see #substitutionHeadNamespace()
     */
    String substitutionHeadName() default "";

    /**
     * Default value of this element.
     *
     * <p>
     * The <pre>'\u0000'</pre> value specified as a default of this annotation element
     * is used as a poor-man's substitute for null to allow implementations
     * to recognize the 'no default value' state.
     */
    String defaultValue() default "\u0000";

    /**
     * Used in {@link XmlElementDecl#scope()} to
     * signal that the declaration is in the global scope.
     */
    public final class GLOBAL {}
}
```

3.13 XmlElementRef.java

Secuencia 47: javax/xml/bind/annotation/XmlElementRef.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;

/**
 * <p>
 * Maps a JavaBean property to a XML element derived from property's type.
 * <p>
 * <b>Usage</b>
 * <p>
 * <tt>&#64;XmlElementRef</tt> annotation can be used with a
 * JavaBean property or from within {@link XmlElementRefs}
 * <p>
 * This annotation dynamically associates an XML element name with the JavaBean
 * property. When a JavaBean property is annotated with {@link
 * XmlElement}, the XML element name is statically derived from the
 * JavaBean property name. However, when this annotation is used, the
 * XML element name is derived from the instance of the type of the
 * JavaBean property at runtime.
 *
 * <h3> XML Schema substitution group support </h3>
 * XML Schema allows a XML document author to use XML element names
 * that were not statically specified in the content model of a
 * schema using substitution groups. Schema derived code provides
 * support for substitution groups using an <i>element property</i>,
 * (section 5.5.5, "Element Property" of JAXB 2.0 specification). An
 * element property method signature is of the form:
 * <pre>
```

```
*     public void setTerm(JAXBElement<? extends Operator>);
*     public JAXBElement<? extends Operator> getTerm();
* </pre>
* <p>
* An element factory method annotated with  {@link XmlElementDecl} is
* used to create a <tt>JAXBElement</tt> instance, containing an XML
* element name. The presence of &#64;XmlElementRef annotation on an
* element property indicates that the element name from <tt>JAXBElement</tt>
* instance be used instead of deriving an XML element name from the
* JavaBean property name.
*
* <p>
* The usage is subject to the following constraints:
* <ul>
*   <li> If the collection item type (for collection property) or
*        property type (for single valued property) is
*        {@link javax.xml.bind.JAXBElement}, then
*        <tt>&#64;XmlElementRef}.name()</tt> and <tt>&#64;XmlElementRef.namespace()</tt> must
*        point an element factory method  with an @XmlElementDecl
*        annotation in a class annotated  with @XmlRegistry (usually
*        ObjectFactory class generated by  the schema compiler) :
*        <ul>
*            <li> @XmlElementDecl.name() must equal @XmlElementRef.name()  </li>
*            <li> @XmlElementDecl.namespace() must equal @XmlElementRef.namespace(). </li>
*        </ul>
*   </li>
*   <li> If the collection item type (for collection property) or
*        property type  (for single valued property) is not
*        {@link javax.xml.bind.JAXBElement}, then the type referenced by the
*        property or field must be annotated  with {@link XmlRootElement}. </li>
*   <li> This annotation can be used with the following annotations:
*        {@link XmlElementWrapper}, {@link XmlJavaTypeAdapter}.
*   </ul>
*
* <p>See "Package Specification" in javax.xml.bind.package javadoc for
* additional common information.</p>
*
* <p><b>Example 1: Ant Task Example</b></p>
* The following Java class hierarchy models an Ant build
* script.  An Ant task corresponds to a class in the class
* hierarchy. The XML element name of an Ant task is indicated by the
* &#64;XmlRootElement annotation on its corresponding class.
* <pre>
*     &#64;XmlRootElement(name="target")
*     class Target {
*         // The presence of &#64;XmlElementRef indicates that the XML
*         // element name will be derived from the &#64;XmlRootElement
*         // annotation on the type (for e.g. "jar" for JarTask).
*         &#64;XmlElementRef
*         List&lt;Task> tasks;
*     }
*
*     abstract class Task {
```

```
*     }
*
*     &#64;XmlRootElement(name="jar")
*     class JarTask extends Task {
*         ...
*     }
*
*     &#64;XmlRootElement(name="javac")
*     class JavacTask extends Task {
*         ...
*     }
*
*     &lt;!-- XML Schema fragment -->
*     &lt;xs:element name="target" type="Target">
*     &lt;xs:complexType name="Target">
*       &lt;xs:sequence>
*         &lt;xs:choice maxOccurs="unbounded">
*           &lt;xs:element ref="jar">
*           &lt;xs:element ref="javac">
*         &lt;/xs:choice>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
*
* </pre>
* <p>
* Thus the following code fragment:
* <pre>
*     Target target = new Target();
*     target.tasks.add(new JarTask());
*     target.tasks.add(new JavacTask());
*     marshal(target);
* </pre>
* will produce the following XML output:
* <pre>
*     &lt;target>
*       &lt;jar>
*         ....
*       &lt;/jar>
*       &lt;javac>
*         ....
*       &lt;/javac>
*     &lt;/target>
* </pre>
* <p>
* It is not an error to have a class that extends <tt>Task</tt>
* that doesn't have {@link XmlRootElement}. But they can't show up in an
* XML instance (because they don't have XML element names).
*
* <p><b>Example 2: XML Schema Susbstitution group support</b>
* <p> The following example shows the annotations for XML Schema
* substitution groups.  The annotations and the ObjectFactory are
* derived from the schema.
*
```

```
* <pre>
*     &#64;XmlElement
*     class Math {
*         //  The value of {@link #type()}is
*         //  JAXBElement.class , which indicates the XML
*         //  element name ObjectFactory - in general a class marked
*         //  with &#64;XmlRegistry. (See ObjectFactory below)
*         //
*         //  The {@link #name()} is "operator", a pointer to a
*         // factory method annotated with a
*         //  {@link XmlElementDecl} with the name "operator". Since
*         //  "operator" is the head of a substitution group that
*         //  contains elements "add" and "sub" elements, "operator"
*         //  element can be substituted in an instance document by
*         //  elements "add" or "sub". At runtime, JAXBElement
*         //  instance contains the element name that has been
*         //  substituted in the XML document.
*         //
*         &#64;XmlElementRef(type=JAXBElement.class,name="operator")
*         JAXBElement&lt;? extends Operator> term;
*     }
*
*     &#64;XmlRegistry
*     class ObjectFactory {
*         &#64;XmlElementDecl(name="operator")
*         JAXBElement&lt;Operator> createOperator(Operator o) {...}
*         &#64;XmlElementDecl(name="add",substitutionHeadName="operator")
*         JAXBElement&lt;Operator> createAdd(Operator o) {...}
*         &#64;XmlElementDecl(name="sub",substitutionHeadName="operator")
*         JAXBElement&lt;Operator> createSub(Operator o) {...}
*     }
*
*     class Operator {
*         ...
*     }
* </pre>
* <p>
* Thus, the following code fragment
* <pre>
*     Math m = new Math();
*     m.term = new ObjectFactory().createAdd(new Operator());
*     marshal(m);
* </pre>
* will produce the following XML output:
* <pre>
*     &lt;math>
*       &lt;add>...&lt;/add>
*     &lt;/math>
* </pre>
*
*
* @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems,Inc. </li><li>Sekhar Vajjhala, Sun
    Microsystems, Inc.</li></ul>
```

```
 * @see XmlElementRefs
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD})
public @interface XmlElementRef {
    /**
     * The Java type being referenced.
     * <p>
     * If the value is DEFAULT.class, the type is inferred from the
     * the type of the JavaBean property.
     */
    Class type() default DEFAULT.class;

    /**
     * This parameter and {@link #name()} are used to determine the
     * XML element for the JavaBean property.
     *
     * <p> If <tt>type()</tt> is <tt>JAXBElement.class</tt> , then
     * <tt>namespace()</tt> and <tt>name()</tt>
     * point to a factory method with {@link XmlElementDecl}. The XML
     * element name is the element name from the factory method's
     * {@link XmlElementDecl} annotation or if an element from its
     * substitution group (of which it is a head element) has been
     * substituted in the XML document, then the element name is from the
     * {@link XmlElementDecl} on the substituted element.
     *
     * <p> If {@link #type()} is not <tt>JAXBElement.class</tt>, then
     * the XML element name is the XML element name statically
     * associated with the type using the annotation {@link
     * XmlRootElement} on the type. If the type is not annotated with
     * an {@link XmlElementDecl}, then it is an error.
     *
     * <p> If <tt>type()</tt> is not <tt>JAXBElement.class</tt>, then
     * this value must be "".
     *
     */
    String namespace() default "";
    /**
     *
     * @see #namespace()
     */
    String name() default "##default";

    /**
     * Used in {@link XmlElementRef#type()} to
     * signal that the type be inferred from the signature
     * of the property.
     */
    static final class DEFAULT {}

    /**
     * Customize the element declaration to be required.
```

```
     * <p>
     * If required() is true, then Javabean property is mapped to
     * an XML schema element declaration with minOccurs="1".
     * maxOccurs is "1" for a single valued property and "unbounded"
     * for a multivalued property.
     *
     * <p>
     * If required() is false, then the Javabean property is mapped
     * to XML Schema element declaration with minOccurs="0".
     * maxOccurs is "1" for a single valued property and "unbounded"
     * for a multivalued property.
     *
     * <p>
     * For compatibility with JAXB 2.1, this property defaults to <tt>true</tt>,
     * despite the fact that {@link XmlElement#required()} defaults to false.
     *
     * @since 2.2
     */
    boolean required() default true;
}
```

3.14 XmlElementRefs.java

Secuencia 48: javax/xml/bind/annotation/XmlElementRefs.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import java.lang.annotation.Retention;
```

```
import java.lang.annotation.Target;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * Marks a property that refers to classes with {@link XmlElement}
 * or JAXBElement.
 *
 * <p>
 * Compared to an element property (property with {@link XmlElement}
 * annotation), a reference property has a different substitution semantics.
 * When a sub-class is assigned to a property, an element property produces
 * the same tag name with @xsi:type, whereas a reference property produces
 * a different tag name (the tag name that's on the the sub-class.)
 *
 * <p> This annotation can be used with the following annotations:
 * {@link XmlJavaTypeAdapter}, {@link XmlElementWrapper}.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Sekhar Vajjhala, Sun
     Microsystems, Inc.</li></ul>
 *
 * @see XmlElementWrapper
 * @see XmlElementRef
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD})
public @interface XmlElementRefs {
    XmlElementRef[] value();
}
```

3.15 XmlElementWrapper.java

Secuencia 49: javax/xml/bind/annotation/XmlElementWrapper.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

/**
 * Generates a wrapper element around XML representation.
 *
 * This is primarily intended to be used to produce a wrapper
 * XML element around collections. The annotation therefore supports
 * two forms of serialization shown below.
 *
 * <pre>
 *    //Example: code fragment
 *      int[] names;
 *
 *    // XML Serialization Form 1 (Unwrapped collection)
 *    &lt;names> ... &lt;/names>
 *    &lt;names> ... &lt;/names>
 *
 *    // XML Serialization Form 2 ( Wrapped collection )
 *    &lt;wrapperElement>
 *       &lt;names> value-of-item &lt;/names>
 *       &lt;names> value-of-item &lt;/names>
 *       ....
 *    &lt;/wrapperElement>
 * </pre>
 *
 * <p> The two serialized XML forms allow a null collection to be
 * represented either by absence or presence of an element with a
 * nillable attribute.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * The <tt>@XmlElementWrapper</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li> JavaBean property </li>
 *   <li> non static, non transient field </li>
 * </ul>
 *
 * <p>The usage is subject to the following constraints:
```

```
 * <ul>
 *   <li> The property must be a collection property </li>
 *   <li> This annotation can be used with the following annotations:
 *           {@link XmlElement},
 *           {@link XmlElements},
 *           {@link XmlElementRef},
 *           {@link XmlElementRefs},
 *           {@link XmlJavaTypeAdapter}.</li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Sekhar Vajjhala, Sun
     Microsystems, Inc.</li></ul>
 * @see XmlElement
 * @see XmlElements
 * @see XmlElementRef
 * @see XmlElementRefs
 * @since JAXB2.0
 *
 */

@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlElementWrapper {
    /**
     * Name of the XML wrapper element. By default, the XML wrapper
     * element name is derived from the JavaBean property name.
     */
    String name() default "##default";

    /**
     * XML target namespace of the XML wrapper element.
     * <p>
     * If the value is "##default", then the namespace is determined
     * as follows:
     * <ol>
     *  <li>
     *  If the enclosing package has {@link XmlSchema} annotation,
     *  and its {@link XmlSchema#elementFormDefault() elementFormDefault}
     *  is {@link XmlNsForm#QUALIFIED QUALIFIED}, then the namespace of
     *  the enclosing class.
     *
     *  <li>
     *  Otherwise "" (which produces unqualified element in the default
     *  namespace.
     * </ol>
     */
    String namespace() default "##default";

    /**
     * If true, the absence of the collection is represented by
     * using <tt>xsi:nil='true'</tt>. Otherwise, it is represented by
```

```
     * the absence of the element.
     */
    boolean nillable() default false;

    /**
     * Customize the wrapper element declaration to be required.
     *
     * <p>
     * If required() is true, then the corresponding generated
     * XML schema element declaration will have <tt>minOccurs="1"</tt>,
     * to indicate that the wrapper element is always expected.
     *
     * <p>
     * Note that this only affects the schema generation, and
     * not the unmarshalling or marshalling capability. This is
     * simply a mechanism to let users express their application constraints
     * better.
     *
     * @since JAXB 2.1
     */
    boolean required() default false;
}
```

3.16 XmlElements.java

Secuencia 50: javax/xml/bind/annotation/XmlElements.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

```

```
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

/**
 * <p>
 * A container for multiple @{@link XmlElement} annotations.
 *
 * Multiple annotations of the same type are not allowed on a program
 * element. This annotation therefore serves as a container annotation
 * for multiple &#64;XmlElements as follows:
 *
 * <pre>
 * &#64;XmlElements({ @XmlElement(...),@XmlElement(...) })
 * </pre>
 *
 * <p>The <tt>@XmlElements</tt> annnotation can be used with the
 * following program elements: </p>
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> non static, non transient field </li>
 * </ul>
 *
 * This annotation is intended for annotation a JavaBean collection
 * property (e.g. List).
 *
 * <p><b>Usage</b></p>
 *
 * <p>The usage is subject to the following constraints:
 * <ul>
 *   <li> This annotation can be used with the following
 *        annotations: @{@link XmlIDREF}, @{@link XmlElementWrapper}. </li>
 *   <li> If @XmlIDREF is also specified on the JavaBean property,
 *        then each &#64;XmlElement.type() must contain a JavaBean
 *        property annotated with <tt>&#64;XmlID</tt>.</li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <hr>
 *
 * <p><b>Example 1:</b> Map to a list of elements</p>
 * <pre>
 *
 *    // Mapped code fragment
 *    public class Foo {
 *        &#64;XmlElements(
 *            &#64;XmlElement(name="A", type=Integer.class),
 *            &#64;XmlElement(name="B", type=Float.class)
```

```
*         }
*         public List items;
*    }
*
*    &lt;!-- XML Representation for a List of {1,2.5}
*            XML output is not wrapped using another element -->
*    ...
*    &lt;A> 1 &lt;/A>
*    &lt;B> 2.5 &lt;/B>
*    ...
*
*    &lt;!-- XML Schema fragment -->
*    &lt;xs:complexType name="Foo">
*      &lt;xs:sequence>
*        &lt;xs:choice minOccurs="0" maxOccurs="unbounded">
*          &lt;xs:element name="A" type="xs:int"/>
*          &lt;xs:element name="B" type="xs:float"/>
*        &lt;xs:choice>
*      &lt;/xs:sequence>
*    &lt;/xs:complexType>
*
* </pre>
*
* <p><b>Example 2:</b> Map to a list of elements wrapped with another element
* </p>
* <pre>
*
*    // Mapped code fragment
*    public class Foo {
*        &#64;XmlElementWrapper(name="bar")
*        &#64;XmlElements(
*            &#64;XmlElement(name="A", type=Integer.class),
*            &#64;XmlElement(name="B", type=Float.class)
*        }
*        public List items;
*    }
*
*    &lt;!-- XML Schema fragment -->
*    &lt;xs:complexType name="Foo">
*      &lt;xs:sequence>
*        &lt;xs:element name="bar">
*          &lt;xs:complexType>
*            &lt;xs:choice minOccurs="0" maxOccurs="unbounded">
*              &lt;xs:element name="A" type="xs:int"/>
*              &lt;xs:element name="B" type="xs:float"/>
*            &lt;/xs:choice>
*          &lt;/xs:complexType>
*        &lt;/xs:element>
*      &lt;/xs:sequence>
*    &lt;/xs:complexType>
* </pre>
*
* <p><b>Example 3:</b> Change element name based on type using an adapter.
```

```
 * </p>
 * <pre>
 *    class Foo {
 *       &#64;XmlJavaTypeAdapter(QtoPAdapter.class)
 *       &#64;XmlElements({
 *           &#64;XmlElement(name="A",type=PX.class),
 *           &#64;XmlElement(name="B",type=PY.class)
 *       })
 *       Q bar;
 *    }
 *
 *    &#64;XmlType abstract class P {...}
 *    &#64;XmlType(name="PX") class PX extends P {...}
 *    &#64;XmlType(name="PY") class PY extends P {...}
 *
 *    &lt;!-- XML Schema fragment -->
 *    &lt;xs:complexType name="Foo">
 *      &lt;xs:sequence>
 *        &lt;xs:element name="bar">
 *          &lt;xs:complexType>
 *            &lt;xs:choice minOccurs="0" maxOccurs="unbounded">
 *              &lt;xs:element name="A" type="PX"/>
 *              &lt;xs:element name="B" type="PY"/>
 *            &lt;/xs:choice>
 *          &lt;/xs:complexType>
 *        &lt;/xs:element>
 *      &lt;/xs:sequence>
 *    &lt;/xs:complexType>
 * </pre>
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Sekhar Vajjhala, Sun
     Microsystems, Inc.</li></ul>
 * @see XmlElement
 * @see XmlElementRef
 * @see XmlElementRefs
 * @see XmlJavaTypeAdapter
 * @since JAXB2.0
 */
@Retention(RUNTIME) @Target({FIELD,METHOD})
public @interface XmlElements {
    /**
     * Collection of @{@link XmlElement} annotations
     */
    XmlElement[] value();
}
```

3.17 XmlEnum.java

Secuencia 51: javax/xml/bind/annotation/XmlEnum.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import static java.lang.annotation.ElementType.TYPE;
import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * <p>
 * Maps an enum type {@link Enum} to XML representation.
 *
 * <p>This annotation, together with {@link XmlEnumValue} provides a
 * mapping of enum type to XML representation.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * The <tt>@XmlEnum</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li>enum type</li>
 * </ul>
 *
 * <p> The usage is subject to the following constraints:
 * <ul>
 *   <li> This annotation can be used the following other annotations:
 *        {@link XmlType},
 *        {@link XmlRootElement} </li>
 * </ul>
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information </p>
 *
 * <p>An enum type is mapped to a schema simple type with enumeration
```

```
 * facets. The schema type is derived from the Java type to which
 * <tt>@XmlEnum.value()</tt>. Each enum constant <tt>@XmlEnumValue</tt>
 * must have a valid lexical representation for the type
 * <tt>@XmlEnum.value()</tt> .
 *
 * <p><b>Examples:</b> See examples in {@link XmlEnumValue}
 *
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({TYPE})
public @interface XmlEnum {
    /**
     * Java type that is mapped to a XML simple type.
     *
     */
    Class<?> value() default String.class;
}
```

3.18 XmlEnumValue.java

Secuencia 52: javax/xml/bind/annotation/XmlEnumValue.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
```

```

/**
 * Maps an enum constant in {@link Enum} type to XML representation.
 *
 * <p> <b>Usage</b> </p>
 *
 * <p> The <tt>@XmlEnumValue</tt> annotation can be used with the
 *     following program elements:
 * <ul>
 *   <li>enum constant</li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p>This annotation, together with {@link XmlEnum} provides a
 * mapping of enum type to XML representation.
 *
 * <p>An enum type is mapped to a schema simple type with enumeration
 * facets. The schema type is derived from the Java type specified in
 * <tt>@XmlEnum.value()</tt>. Each enum constant <tt>@XmlEnumValue</tt>
 * must have a valid lexical representation for the type
 * <tt>@XmlEnum.value()</tt>
 *
 * <p> In the absence of this annotation, {@link Enum#name()} is used
 * as the XML representation.
 *
 * <p> <b>Example 1: </b>Map enum constant name -> enumeration facet</p>
 * <pre>
 *     //Example: Code fragment
 *     &#64;XmlEnum(String.class)
 *     public enum Card { CLUBS, DIAMONDS, HEARTS, SPADES }
 *
 *     &lt;!-- Example: XML Schema fragment -->
 *     &lt;xs:simpleType name="Card">
 *       &lt;xs:restriction base="xs:string"/>
 *         &lt;xs:enumeration value="CLUBS"/>
 *         &lt;xs:enumeration value="DIAMONDS"/>
 *         &lt;xs:enumeration value="HEARTS"/>
 *         &lt;xs:enumeration value="SPADES"/>
 *     &lt;/xs:simpleType>
 * </pre>
 *
 * <p><b>Example 2: </b>Map enum constant name(value) -> enumeration facet </p>
 * <pre>
 *     //Example: code fragment
 *     &#64;XmlType
 *     &#64;XmlEnum(Integer.class)
 *     public enum Coin {
 *         &#64;XmlEnumValue("1") PENNY(1),
 *         &#64;XmlEnumValue("5") NICKEL(5),
 *         &#64;XmlEnumValue("10") DIME(10),
 *         &#64;XmlEnumValue("25") QUARTER(25) }
```

```
 *
 *     &lt;!-- Example: XML Schema fragment -->
 *     &lt;xs:simpleType name="Coin">
 *       &lt;xs:restriction base="xs:int">
 *         &lt;xs:enumeration value="1"/>
 *         &lt;xs:enumeration value="5"/>
 *         &lt;xs:enumeration value="10"/>
 *         &lt;xs:enumeration value="25"/>
 *       &lt;/xs:restriction>
 *     &lt;/xs:simpleType>
 * </pre>
 *
 * <p><b>Example 3: </b>Map enum constant name -> enumeration facet </p>
 *
 * <pre>
 *     //Code fragment
 *     &#64;XmlType
 *     &#64;XmlEnum(Integer.class)
 *     public enum Code {
 *         &#64;XmlEnumValue("1") ONE,
 *         &#64;XmlEnumValue("2") TWO;
 *     }
 *
 *     &lt;!-- Example: XML Schema fragment -->
 *     &lt;xs:simpleType name="Code">
 *       &lt;xs:restriction base="xs:int">
 *         &lt;xs:enumeration value="1"/>
 *         &lt;xs:enumeration value="2"/>
 *       &lt;/xs:restriction>
 *     &lt;/xs:simpleType>
 * </pre>
 *
 * @since JAXB 2.0
 */
@Retention(RUNTIME)
@Target({FIELD})
public @interface XmlEnumValue {
    String value();
}
```

3.19 XmlID.java

Secuencia 53: javax/xml/bind/annotation/XmlID.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p>
 * Maps a JavaBean property to XML ID.
 *
 * <p>
 * To preserve referential integrity of an object graph across XML
 * serialization followed by a XML deserialization, requires an object
 * reference to be marshalled by reference or containment
 * appropriately. Annotations <tt>&#64;XmlID</tt> and <tt>&#64;XmlIDREF</tt>
 * together allow a customized mapping of a JavaBean property's
 * type by containment or reference.
 *
 * <p><b>Usage</b> </p>
 * The <tt>&#64;XmlID</tt> annotation can be used with the following
 * program elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> non static, non transient field </li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * The usage is subject to the following constraints:
 * <ul>
 *   <li> At most one field or property in a class can be annotated
 *        with <tt>&#64;XmlID</tt>.  </li>
 *   <li> The JavaBean property's type must be <tt>java.lang.String</tt>.</li>
 *   <li> The only other mapping annotations that can be used
 *        with <tt>&#64;XmlID</tt>
 *        are:<tt>&#64;XmlElement</tt> and <tt>&#64;XmlAttribute</tt>.</li>
```

```
 * </ul>
 *
 * <p><b>Example</b>: Map a JavaBean property's type to <tt>xs:ID</tt></p>
 * <pre>
 *    // Example: code fragment
 *    public class Customer {
 *        &#64;XmlAttribute
 *        &#64;XmlID
 *        public String getCustomerID();
 *        public void setCustomerID(String id);
 *        .... other properties not shown
 *    }
 *
 *    &lt;!-- Example: XML Schema fragment -->
 *    &lt;xs:complexType name="Customer">
 *      &lt;xs:complexContent>
 *        &lt;xs:sequence>
 *          ....
 *        &lt;/xs:sequence>
 *        &lt;xs:attribute name="customerID" type="xs:ID"/>
 *      &lt;/xs:complexContent>
 *    &lt;/xs:complexType>
 * </pre>
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @see XmlIDREF
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlID { }
```

3.20 XmlIDREF.java

Secuencia 54: javax/xml/bind/annotation/XmlIDREF.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p>
 * Maps a JavaBean property to XML IDREF.
 *
 * <p>
 * To preserve referential integrity of an object graph across XML
 * serialization followed by a XML deserialization, requires an object
 * reference to be marshalled by reference or containment
 * appropriately. Annotations <tt>&#64;XmlID</tt> and <tt>&#64;XmlIDREF</tt>
 * together allow a customized mapping of a JavaBean property's
 * type by containment or reference.
 *
 * <p><b>Usage</b> </p>
 * The <tt>&#64;XmlIDREF</tt> annotation can be used with the following
 * program elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> non static, non transient field </li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p> The usage is subject to the following constraints:
 * <ul>
 *
 *   <li> If the type of the field or property is a collection type,
 *        then the collection item type must contain a property or
 *        field annotated with <tt>&#64;XmlID</tt>.  </li>
 *   <li> If the field or property is single valued, then the type of
 *        the property or field must contain a property or field
 *        annotated with <tt>&#64;XmlID</tt>.
 *        <p>Note: If the collection item type or the type of the
 *        property (for non collection type) is java.lang.Object, then
 *        the instance must contain a property/field annotated with
 *        <tt>&#64;XmlID</tt> attribute.
 *        </li>
 *   <li> This annotation can be used with the following annotations:
 *        {@link XmlElement}, {@link XmlAttribute}, {@link XmlList},
```

```
*       and {@link XmlElements}.</li>
*
* </ul>
* <p><b>Example:</b> Map a JavaBean property to <tt>xs:IDREF</tt>
*   (i.e. by reference rather than by containment)</p>
* <pre>
*
*   //EXAMPLE: Code fragment
*   public class Shipping {
*       &#64;XmlIDREF public Customer getCustomer();
*       public void setCustomer(Customer customer);
*       ....
*    }
*
*   &lt;!-- Example: XML Schema fragment -->
*   &lt;xs:complexType name="Shipping">
*     &lt;xs:complexContent>
*       &lt;xs:sequence>
*         &lt;xs:element name="customer" type="xs:IDREF"/>
*         ....
*       &lt;/xs:sequence>
*     &lt;/xs:complexContent>
*   &lt;/xs:complexType>
*
* </pre>
*
*
* <p><b>Example 2: </b> The following is a complete example of
* containment versus reference.
*
* <pre>
*    // By default, Customer maps to complex type <tt>xs:Customer</tt>
*    public class Customer {
*
*        // map JavaBean property type to <tt>xs:ID</tt>
*        &#64;XmlID public String getCustomerID();
*        public void setCustomerID(String id);
*
*        // .... other properties not shown
*    }
*
*
*   // By default, Invoice maps to a complex type <tt>xs:Invoice</tt>
*   public class Invoice {
*
*       // map by reference
*       &#64;XmlIDREF public Customer getCustomer();
*       public void setCustomer(Customer customer);
*
*      // .... other properties not shown here
*   }
*
*   // By default, Shipping maps to complex type <tt>xs:Shipping</tt>
```

```
*   public class Shipping {
*
*       // map by reference
*       &#64;XmlIDREF public Customer getCustomer();
*       public void setCustomer(Customer customer);
*   }
*
*   // at least one class must reference Customer by containment;
*   // Customer instances won't be marshalled.
*   &#64;XmlElement(name="CustomerData")
*   public class CustomerData {
*       // map reference to Customer by containment by default.
*       public Customer getCustomer();
*
*       // maps reference to Shipping by containment by default.
*       public Shipping getShipping();
*
*       // maps reference to Invoice by containment by default.
*       public Invoice getInvoice();
*   }
*
*   &lt;!-- XML Schema mapping for above code frament -->
*
*   &lt;xs:complexType name="Invoice">
*     &lt;xs:complexContent>
*       &lt;xs:sequence>
*         &lt;xs:element name="customer" type="xs:IDREF"/>
*         ....
*       &lt;/xs:sequence>
*     &lt;/xs:complexContent>
*   &lt;/xs:complexType>
*
*   &lt;xs:complexType name="Shipping">
*     &lt;xs:complexContent>
*       &lt;xs:sequence>
*         &lt;xs:element name="customer" type="xs:IDREF"/>
*         ....
*       &lt;/xs:sequence>
*     &lt;/xs:complexContent>
*   &lt;/xs:complexType>
*
*   &lt;xs:complexType name="Customer">
*     &lt;xs:complexContent>
*       &lt;xs:sequence>
*         ....
*       &lt;/xs:sequence>
*       &lt;xs:attribute name="CustomerID" type="xs:ID"/>
*     &lt;/xs:complexContent>
*   &lt;/xs:complexType>
*
*   &lt;xs:complexType name="CustomerData">
*     &lt;xs:complexContent>
*       &lt;xs:sequence>
```

```
*         &lt;xs:element name="customer" type="xs:Customer"/>
*         &lt;xs:element name="shipping" type="xs:Shipping"/>
*         &lt;xs:element name="invoice"  type="xs:Invoice"/>
*       &lt;/xs:sequence>
*     &lt;/xs:complexContent>
*   &lt;/xs:complexType>
*
*   &lt;xs:element name"customerData" type="xs:CustomerData"/>
*
*   &lt;!-- Instance document conforming to the above XML Schema -->
*    &lt;customerData>
*       &lt;customer customerID="Alice">
*           ....
*       &lt;/customer>
*
*       &lt;shipping customer="Alice">
*           ....
*       &lt;/shipping>
*
*       &lt;invoice customer="Alice">
*           ....
*       &lt;/invoice>
*   &lt;/customerData>
*
* </pre>
*
* <p><b>Example 3: </b> Mapping List to repeating element of type IDREF
* <pre>
*     // Code fragment
*     public class Shipping {
*         &#64;XmlIDREF
*         &#64;XmlElement(name="Alice")
*             public List customers;
*     }
*
*     &lt;!-- XML schema fragment -->
*     &lt;xs:complexType name="Shipping">
*       &lt;xs:sequence>
*         &lt;xs:choice minOccurs="0" maxOccurs="unbounded">
*           &lt;xs:element name="Alice" type="xs:IDREF"/>
*         &lt;/xs:choice>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
* </pre>
*
* <p><b>Example 4: </b> Mapping a List to a list of elements of type IDREF.
* <pre>
*     //Code fragment
*     public class Shipping {
*         &#64;XmlIDREF
*         &#64;XmlElements(
*             &#64;XmlElement(name="Alice", type="Customer.class")
*              &#64;XmlElement(name="John", type="InternationalCustomer.class")
```

```
 *         public List customers;
 *     }
 *
 *     &lt;!-- XML Schema fragment -->
 *     &lt;xs:complexType name="Shipping">
 *       &lt;xs:sequence>
 *         &lt;xs:choice minOccurs="0" maxOccurs="unbounded">
 *           &lt;xs:element name="Alice" type="xs:IDREF"/>
 *           &lt;xs:element name="John" type="xs:IDREF"/>
 *         &lt;/xs:choice>
 *       &lt;/xs:sequence>
 *     &lt;/xs:complexType>
 * </pre>
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @see XmlID
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlIDREF {}
```

3.21 XmlInlineBinaryData.java

Secuencia 55: javax/xml/bind/annotation/XmlInlineBinaryData.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;
```

```
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.TYPE;

import javax.xml.transform.Source;
import javax.xml.bind.attachment.AttachmentMarshaller;
import javax.activation.DataHandler;

/**
 * Disable consideration of XOP encoding for datatypes that are bound to
 * base64-encoded binary data in XML.
 *
 * <p>
 * When XOP encoding is enabled as described in {@link AttachmentMarshaller#isXOPPackage()}, this
     annotation disables datatypes such as {@link java.awt.Image} or {@link Source} or <tt>byte[]</
     tt> that are bound to base64-encoded binary from being considered for
 * XOP encoding. If a JAXB property is annotated with this annotation or if
 * the JAXB property's base type is annotated with this annotation,
 * neither
 * {@link AttachmentMarshaller#addMtomAttachment(DataHandler, String, String)}
 * nor
 * {@link AttachmentMarshaller#addMtomAttachment(byte[], int, int, String, String, String)} is
 * ever called for the property. The binary data will always be inlined.
 *
 * @author Joseph Fialli
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD,TYPE})
public @interface XmlInlineBinaryData {
}
```

3.22 XmlList.java

Secuencia 56: javax/xml/bind/annotation/XmlList.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.PARAMETER;

/**
 * Used to map a property to a list simple type.
 *
 * <p><b>Usage</b> </p>
 * <p>
 * The <tt>@XmlList</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li> JavaBean property </li>
 *   <li> field </li>
 * </ul>
 *
 * <p>
 * When a collection property is annotated just with @XmlElement,
 * each item in the collection will be wrapped by an element.
 * For example,
 *
 * <pre>
 * &#64;XmlRootElement
 * class Foo {
 *     &#64;XmlElement
 *     List&lt;String> data;
 * }
 * </pre>
 *
 * would produce XML like this:
 *
 * <pre>
 * &lt;foo>
 *   &lt;data>abc</data>
 *   &lt;data>def</data>
 * &lt;/foo>
 * </pre>
 *
 * &#64;XmlList annotation, on the other hand, allows multiple values to be
 * represented as whitespace-separated tokens in a single element. For example,
```

```
 *
 * <pre>
 * &#64;XmlRootElement
 * class Foo {
 *     &#64;XmlElement
 *     &#64;XmlList
 *     List&lt;String> data;
 * }
 * </pre>
 *
 * the above code will produce XML like this:
 *
 * <pre>
 * &lt;foo>
 *   &lt;data>abc def</data>
 * &lt;/foo>
 * </pre>
 *
 * <p>This annotation can be used with the following annotations:
 *        {@link XmlElement},
 *        {@link XmlAttribute},
 *        {@link XmlValue},
 *        {@link XmlIDREF}.
 *  <ul>
 *    <li> The use of <tt>@XmlList</tt> with {@link XmlValue} while
 *         allowed, is redundant since  {@link XmlList} maps a
 *         collection type to a simple schema type that derives by
 *         list just as {@link XmlValue} would. </li>
 *
 *    <li> The use of <tt>@XmlList</tt> with {@link XmlAttribute} while
 *         allowed, is redundant since  {@link XmlList} maps a
 *         collection type to a simple schema type that derives by
 *         list just as {@link XmlAttribute} would. </li>
 *  </ul>
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Sekhar Vajjhala, Sun
     Microsystems, Inc.</li></ul>
 * @since JAXB2.0
 */
@Retention(RUNTIME) @Target({FIELD,METHOD,PARAMETER})
public @interface XmlList {
}
```

3.23 XmlMimeType.java

Secuencia 57: javax/xml/bind/annotation/XmlMimeType.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.PARAMETER;

import javax.xml.transform.Source;

/**
 * Associates the MIME type that controls the XML representation of the property.
 *
 * <p>
 * This annotation is used in conjunction with datatypes such as
 * {@link java.awt.Image} or {@link Source} that are bound to base64-encoded binary in XML.
 *
 * <p>
 * If a property that has this annotation has a sibling property bound to
 * the xmime:contentType attribute, and if in the instance the property has a value,
 * the value of the attribute takes precedence and that will control the marshalling.
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD,PARAMETER})
public @interface XmlMimeType {
    /**
     * The textual representation of the MIME type,
     * such as "image/jpeg" "image/*", "text/xml; charset=iso-8859-1" and so on.
     */
    String value();
}
```

3.24 XmlMixed.java

Secuencia 58: javax/xml/bind/annotation/XmlMixed.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;

import org.w3c.dom.Element;
import javax.xml.bind.JAXBElement;

/**
 * <p>
 * Annotate a JavaBean multi-valued property to support mixed content.
 *
 * <p>
 * The usage is subject to the following constraints:
 * <ul>
 *   <li> can be used with &#64;XmlElementRef, &#64;XmlElementRefs or &#64;XmlAnyElement</li>
 * </ul>
 * <p>
 * The following can be inserted into &#64;XmlMixed annotated multi-valued property
 * <ul>
 * <li>XML text information items are added as values of java.lang.String.</li>
```

```
* <li>Children element information items are added as instances of
* {@link JAXBElement} or instances with a class that is annotated with
* &#64;XmlRootElement.</li>
* <li>Unknown content that is not be bound to a JAXB mapped class is inserted
* as {@link Element}. (Assumes property annotated with &#64;XmlAnyElement)</li>
* </ul>
*
* Below is an example of binding and creation of mixed content.
* <pre>
*  &lt;!-- schema fragment having  mixed content -->
*  &lt;xs:complexType name="letterBody" mixed="true">
*    &lt;xs:sequence>
*      &lt;xs:element name="name" type="xs:string"/>
*      &lt;xs:element name="quantity" type="xs:positiveInteger"/>
*      &lt;xs:element name="productName" type="xs:string"/>
*      &lt;!-- etc. -->
*    &lt;/xs:sequence>
*  &lt;/xs:complexType>
*  &lt;xs:element name="letterBody" type="letterBody"/>
*
* // Schema-derived Java code:
* // (Only annotations relevant to mixed content are shown below,
* //  others are ommitted.)
* import java.math.BigInteger;
* public class ObjectFactory {
*      // element instance factories
*      JAXBElement&lt;LetterBody> createLetterBody(LetterBody value);
*      JAXBElement&lt;String>     createLetterBodyName(String value);
*      JAXBElement&lt;BigInteger> createLetterBodyQuantity(BigInteger value);
*      JAXBElement&lt;String>     createLetterBodyProductName(String value);
*      // type instance factory
*      LetterBody> createLetterBody();
* }
* </pre>
* <pre>
* public class LetterBody {
*      // Mixed content can contain instances of Element classes
*      // Name, Quantity and ProductName. Text data is represented as
*      // java.util.String for text.
*      &#64;XmlMixed
*      &#64;XmlElementRefs({
*              &#64;XmlElementRef(name="productName", type=JAXBElement.class),
*              &#64;XmlElementRef(name="quantity", type=JAXBElement.class),
*              &#64;XmlElementRef(name="name", type=JAXBElement.class)})
*      List getContent(){...}
* }
* </pre>
* The following is an XML instance document with mixed content
* <pre>
* &lt;letterBody>
* Dear Mr.&lt;name>Robert Smith&lt;/name>
* Your order of &lt;quantity>1&lt;/quantity> &lt;productName>Baby
* Monitor&lt;/productName> shipped from our warehouse. ....
```

```
 * &lt;/letterBody>
 * </pre>
 * that can be constructed using following JAXB API calls.
 * <pre>
 * LetterBody lb = ObjectFactory.createLetterBody();
 * JAXBElement&lt;LetterBody> lbe = ObjectFactory.createLetterBody(lb);
 * List gcl = lb.getContent();  //add mixed content to general content property.
 * gcl.add("Dear Mr.");  // add text information item as a String.
 *
 * // add child element information item
 * gcl.add(ObjectFactory.createLetterBodyName("Robert Smith"));
 * gcl.add("Your order of "); // add text information item as a String
 *
 * // add children element information items
 * gcl.add(ObjectFactory.
 *                      createLetterBodyQuantity(new BigInteger("1")));
 * gcl.add(ObjectFactory.createLetterBodyProductName("Baby Monitor"));
 * gcl.add("shipped from our warehouse");  // add text information item
 * </pre>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 * @author Kohsuke Kawaguchi
 * @since JAXB2.0
 */
@Retention(RUNTIME)
@Target({FIELD,METHOD})
public @interface XmlMixed {
}
```

3.25 XmlNs.java

Secuencia 59: javax/xml/bind/annotation/XmlNs.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * <p>
 * Associates a namespace prefix with a XML namespace URI.
 *
 * <p><b>Usage</b></p>
 * <p><tt>@XmlNs</tt> annotation is intended for use from other
 * program annotations.
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p><b>Example:</b>See <tt>XmlSchema</tt> annotation type for an example.
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({})
public @interface XmlNs {
    /**
     * Namespace prefix
     */
    String prefix();

    /**
     * Namespace URI
     */
    String namespaceURI();
}
```

3.26 XmlNsForm.java

Secuencia 60: javax/xml/bind/annotation/XmlNsForm.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

/**
 * Enumeration of XML Schema namespace qualifications.
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p><b>Usage</b>
 * <p>
 * The namespace qualification values are used in the annotations
 * defined in this packge. The enumeration values are mapped as follows:
 *
 * <p>
 * <table border="1" cellpadding="4" cellspacing="3">
 *   <tbody>
 *     <tr>
 *       <td><b>Enum Value</b></td>
 *       <td><b>XML Schema Value</b></td>
 *     </tr>
 *
 *     <tr valign="top">
 *       <td>UNQUALIFIED</td>
 *       <td>unqualified</td>
 *     </tr>
 *     <tr valign="top">
 *       <td>QUALIFIED</td>
 *       <td>qualified</td>
 *     </tr>
 *     <tr valign="top">
 *       <td>UNSET</td>
 *       <td>namespace qualification attribute is absent from the
 *           XML Schema fragment</td>
 *     </tr>
 *   </tbody>
 * </table>
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
```

```
 * @since JAXB2.0
 */
public enum XmlNsForm {UNQUALIFIED, QUALIFIED, UNSET}
```

3.27 XmlRegistry.java

Secuencia 61: javax/xml/bind/annotation/XmlRegistry.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * Marks a class that has {@link XmlElementDecl}s.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Sekhar Vajjhala, Sun
     Microsystems, Inc.</li></ul>
 * @since JAXB 2.0
 * @see XmlElementDecl
 */
@Retention(RUNTIME)
@Target({TYPE})
public @interface XmlRegistry {
}
```

3.28 XmlRootElement.java

Secuencia 62: javax/xml/bind/annotation/XmlRootElement.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.TYPE;

/**
 * Maps a class or an enum type to an XML element.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * The &#64;XmlRootElement annotation can be used with the following program
 * elements:
 * <ul>
 *   <li> a top level class </li>
 *   <li> an enum type </li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p>
 * When a top level class or an enum type is annotated with the
 * &#64;XmlRootElement annotation, then its value is represented
 * as XML element in an XML document.
```

```
*
* <p> This annotation can be used with the following annotations:
* {@link XmlType}, {@link XmlEnum}, {@link XmlAccessorType},
* {@link XmlAccessorOrder}.
* <p>

* <p>
* <b>Example 1: </b> Associate an element with XML Schema type
* <pre>
*     // Example: Code fragment
*     &#64;XmlRootElement
*     class Point {
*        int x;
*        int y;
*        Point(int _x,int _y) {x=_x;y=_y;}
*     }
* </pre>
*
* <pre>
*     //Example: Code fragment corresponding to XML output
*     marshal( new Point(3,5), System.out);
* </pre>
*
* <pre>
*     &lt;!-- Example: XML output -->
*     &lt;point>
*       &lt;x> 3 </x>
*       &lt;y> 5 </y>
*     &lt;/point>
* </pre>
*
* The annotation causes an global element declaration to be produced
* in the schema. The global element declaration is associated with
* the XML schema type to which the class is mapped.
*
* <pre>
*     &lt;!-- Example: XML schema definition -->
*     &lt;xs:element name="point" type="point"/>
*     &lt;xs:complexType name="point">
*       &lt;xs:sequence>
*         &lt;xs:element name="x" type="xs:int"/>
*         &lt;xs:element name="y" type="xs:int"/>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
* </pre>
*
* <p>
*
* <b>Example 2: Orthogonality to type inheritance </b>
*
* <p>
* An element declaration annotated on a type is not inherited by its
* derived types. The following example shows this.
```

```
* <pre>
*     // Example: Code fragment
*     &#64;XmlRootElement
*     class Point3D extends Point {
*         int z;
*         Point3D(int _x,int _y,int _z) {super(_x,_y);z=_z;}
*     }
*
*     //Example: Code fragment corresponding to XML output *
*     marshal( new Point3D(3,5,0), System.out );
*
*     &lt;!-- Example: XML output -->
*     &lt;!-- The element name is point3D not point -->
*     &lt;point3D>
*       &lt;x>3&lt;/x>
*       &lt;y>5&lt;/y>
*       &lt;z>0&lt;/z>
*     &lt;/point3D>
*
*     &lt;!-- Example: XML schema definition -->
*     &lt;xs:element name="point3D" type="point3D"/>
*     &lt;xs:complexType name="point3D">
*       &lt;xs:complexContent>
*         &lt;xs:extension base="point">
*           &lt;xs:sequence>
*             &lt;xs:element name="z" type="xs:int"/>
*           &lt;/xs:sequence>
*         &lt;/xs:extension>
*       &lt;/xs:complexContent>
*     &lt;/xs:complexType>
* </pre>
*
* <b>Example 3: </b> Associate a global element with XML Schema type
* to which the class is mapped.
* <pre>
*     //Example: Code fragment
*     &#64;XmlRootElement(name="PriceElement")
*     public class USPrice {
*         &#64;XmlElement
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example: XML schema definition -->
*     &lt;xs:element name="PriceElement" type="USPrice"/>
*     &lt;xs:complexType name="USPrice">
*       &lt;xs:sequence>
*         &lt;xs:element name="price" type="xs:decimal"/>
*       &lt;/sequence>
*     &lt;/xs:complexType>
* </pre>
*
* @author Sekhar Vajjhala, Sun Microsystems, Inc.
* @since JAXB2.0
```

```
 */
@Retention(RUNTIME)
@Target({TYPE})
public @interface XmlRootElement {
    /**
     * namespace name of the XML element.
     * <p>
     * If the value is "##default", then the XML namespace name is derived
     * from the package of the class ( {@link XmlSchema} ). If the
     * package is unnamed, then the XML namespace is the default empty
     * namespace.
     */
    String namespace() default "##default";

    /**
     * local name of the XML element.
     * <p>
     * If the value is "##default", then the name is derived from the
     * class name.
     *
     */
    String name() default "##default";

}
```

3.29 XmlSchema.java

Secuencia 63: javax/xml/bind/annotation/XmlSchema.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```java
package javax.xml.bind.annotation;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p> Maps a package name to a XML namespace. </p>
 *
 * <h3>Usage</h3>
 * <p>
 * The XmlSchema annotation can be used with the following program
 * elements:
 * <ul>
 *   <li>package</li>
 * </ul>
 *
 * <p>
 * This is a package level annotation and follows the recommendations
 * and restrictions contained in JSR 175, section III, "Annotations".
 * Thus the usage is subject to the following constraints and
 * recommendations.
 * <ul>
 *   <li> There can only be one package declaration as noted in JSR
 *        175, section III, "Annotations". </li>
 *   <li> JSR 175 recommends package-info.java for package level
 *        annotations. JAXB Providers that follow this recommendation
 *        will allow the package level annotations to be defined in
 *        package-info.java.
 * </ul>
 * <p>
 *
 * <p><b>Example 1:</b> Customize name of XML namespace to which
 * package is mapped.</p>
 *
 * <pre>
 *    &#64;javax.xml.bind.annotation.XmlSchema (
 *      namespace = "http://www.example.com/MYPO1"
 *    )
 *
 *    &lt;!-- XML Schema fragment -->
 *    &lt;schema
 *      xmlns=...
 *      xmlns:po=....
 *      targetNamespace="http://www.example.com/MYPO1"
 *    >
 *    &lt;!-- prefixes generated by default are implementation
 *            depedenent -->
 * </pre>
 *
 * <p><b>Example 2:</b> Customize namespace prefix, namespace URI
```

```
 * mapping</p>
 *
 * <pre>
 *    // Package level annotation
 *    &#64;javax.xml.bind.annotation.XmlSchema (
 *      xmlns = {
 *        &#64;javax.xml.bind.annotation.XmlNs(prefix = "po",
 *                   namespaceURI="http://www.example.com/myPO1"),
 *
 *        &#64;javax.xml.bind.annotation.XmlNs(prefix="xs",
 *                   namespaceURI="http://www.w3.org/2001/XMLSchema")
 *      )
 *    )
 *
 *    &lt;!-- XML Schema fragment -->
 *    &lt;schema
 *        xmlns:xs="http://www.w3.org/2001/XMLSchema"
 *        xmlns:po="http://www.example.com/PO1"
 *        targetNamespace="http://www.example.com/PO1">
 *
 * </pre>
 *
 * <p><b>Example 3:</b> Customize elementFormDefault</p>
 * <pre>
 *    &#64;javax.xml.bind.annotation.XmlSchema (
 *      elementFormDefault=XmlNsForm.UNQUALIFIED
 *      ...
 *    )
 *
 *    &lt;!-- XML Schema fragment -->
 *    &lt;schema
 *        xmlns="http://www.w3.org/2001/XMLSchema"
 *        xmlns:po="http://www.example.com/PO1"
 *        elementFormDefault="unqualified">
 *
 * </pre>

 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target(PACKAGE)
public @interface XmlSchema {

    /**
     * Customize the namespace URI, prefix associations. By default,
     * the namespace prefixes for a XML namespace are generated by a
     * JAXB Provider in an implementation dependent way.
     */
    XmlNs[]  xmlns() default {};

    /**
     * Name of the XML namespace.
```

```
     */
    String namespace() default "";

    /**
     * Namespace qualification for elements. By default, element
     * default attribute will be absent from the XML Schema fragment.
     */
    XmlNsForm elementFormDefault() default XmlNsForm.UNSET;

    /**
     * Namespace qualification for attributes. By default,
     * attributesFormDefault will be absent from the XML Schema fragment.
     */
    XmlNsForm attributeFormDefault() default XmlNsForm.UNSET;

    /**
     * Indicates that this namespace (specified by {@link #namespace()})
     * has a schema already available exeternally, available at this location.
     *
     * <p>
     * This instructs the JAXB schema generators to simply refer to
     * the pointed schema, as opposed to generating components into the schema.
     * This schema is assumed to match what would be otherwise produced
     * by the schema generator (same element names, same type names...)
     *
     * <p>
     * This feature is intended to be used when a set of the Java classes
     * is originally generated from an existing schema, hand-written to
     * match externally defined schema, or the generated schema is modified
     * manually.
     *
     * <p>
     * Value could be any absolute URI, like <tt>http://example.org/some.xsd</tt>.
     * It is also possible to specify the empty string, to indicate
     * that the schema is externally available but the location is
     * unspecified (and thus it's the responsibility of the reader of the generate
     * schema to locate it.) Finally, the default value of this property
     * <tt>"##generate"</tt> indicates that the schema generator is going
     * to generate components for this namespace (as it did in JAXB 2.0.)
     *
     * <p>
     * Multiple {@link XmlSchema} annotations on multiple packages are allowed
     * to govern the same {@link #namespace()}. In such case, all of them
     * must have the same {@link #location()} values.
     *
     *
     * <h3>Note to implementor</h3>
     * <p>
     * More precisely, the value must be either <tt>""</tt>, <tt>"##generate"</tt>, or
     * <a href="http://www.w3.org/TR/xmlschema-2/#anyURI">
     * a valid lexical representation of <tt>xs:anyURI</tt></a> that begins
     * with <tt>&lt;scheme>:</tt>.
     *
```

```
     * <p>
     * A schema generator is expected to generate a corresponding
     * <tt>&lt;xs:import namespace="..." schemaLocation="..."/></tt> (or
     * no <tt>schemaLocation</tt> attribute at all if the empty string is specified.)
     * However, the schema generator is allowed to use a different value in
     * the <tt>schemaLocation</tt> attribute (including not generating
     * such attribute), for example so that the user can specify a local
     * copy of the resource through the command line interface.
     *
     * @since JAXB2.1
     */
    String location() default NO_LOCATION;

    /**
     * The default value of the {@link #location()} attribute,
     * which indicates that the schema generator will generate
     * components in this namespace.
     */
    // the actual value is chosen because ## is not a valid
    // sequence in xs:anyURI.
    static final String NO_LOCATION = "##generate";
}
```

3.30 XmlSchemaType.java

Secuencia 64: javax/xml/bind/annotation/XmlSchemaType.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

```

```
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.PACKAGE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * Maps a Java type to a simple schema built-in type.
 *
 * <p> <b>Usage</b> </p>
 * <p>
 * <tt>@XmlSchemaType</tt> annotation can be used with the following program
 * elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> field </li>
 *   <li> package</li>
 * </ul>
 *
 * <p> <tt>@XmlSchemaType</tt> annotation defined for Java type
 * applies to all references to the Java type from a property/field.
 * A <tt>@XmlSchemaType</tt> annotation specified on the
 * property/field overrides the <tt>@XmlSchemaType</tt> annotation
 * specified at the package level.
 *
 * <p> This annotation can be used with the following annotations:
 * {@link XmlElement},  {@link XmlAttribute}.
 * <p>
 * <b>Example 1: </b> Customize mapping of XMLGregorianCalendar on the
 *  field.
 *
 * <pre>
 *     //Example: Code fragment
 *     public class USPrice {
 *         &#64;XmlElement
 *         &#64;XmlSchemaType(name="date")
 *         public XMLGregorianCalendar date;
 *     }
 *
 *     &lt;!-- Example: Local XML Schema element -->
 *     &lt;xs:complexType name="USPrice"/>
 *       &lt;xs:sequence>
 *         &lt;xs:element name="date" type="xs:date"/>
 *       &lt;/sequence>
 *     &lt;/xs:complexType>
 * </pre>
 *
 * <p> <b> Example 2: </b> Customize mapping of XMLGregorianCalendar at package
 *     level </p>
 * <pre>
 *     package foo;
```

```
 *     &#64;javax.xml.bind.annotation.XmlSchemaType(
 *          name="date", type=javax.xml.datatype.XMLGregorianCalendar.class)
 *     }
 * </pre>
 *
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD,METHOD,PACKAGE})
public @interface XmlSchemaType {
    String name();
    String namespace() default "http://www.w3.org/2001/XMLSchema";
    /**
     * If this annotation is used at the package level, then value of
     * the type() must be specified.
     */

    Class type() default DEFAULT.class;

    /**
     * Used in {@link XmlSchemaType#type()} to
     * signal that the type be inferred from the signature
     * of the property.
     */

    static final class DEFAULT {}

}
```

3.31 XmlSchemaTypes.java

Secuencia 65: javax/xml/bind/annotation/XmlSchemaTypes.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

package javax.xml.bind.annotation;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.PACKAGE;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

/**
 * <p>
 * A container for multiple @{@link XmlSchemaType} annotations.
 *
 * <p> Multiple annotations of the same type are not allowed on a program
 * element. This annotation therefore serves as a container annotation
 * for multiple &#64;XmlSchemaType annotations as follows:
 *
 * <pre>
 * &#64;XmlSchemaTypes({ @XmlSchemaType(...), @XmlSchemaType(...) })
 * </pre>
 * <p>The <tt>@XmlSchemaTypes</tt> annnotation can be used to
 * define {@link XmlSchemaType} for different types at the
 * package level.
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems, Inc.</li></ul>
 * @see XmlSchemaType
 * @since JAXB2.0
 */
@Retention(RUNTIME) @Target({PACKAGE})
public @interface XmlSchemaTypes {
    /**
     * Collection of @{@link XmlSchemaType} annotations
     */
    XmlSchemaType[] value();
}
```

3.32 XmlSeeAlso.java

Secuencia 66: javax/xml/bind/annotation/XmlSeeAlso.java

```
/*
 * Copyright (c) 2006, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import javax.xml.bind.JAXBContext;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * Instructs JAXB to also bind other classes when binding this class.
 *
 * <p>
 * Java makes it impractical/impossible to list all sub-classes of
 * a given class. This often gets in a way of JAXB users, as it JAXB
 * cannot automatically list up the classes that need to be known
 * to {@link JAXBContext}.
 *
 * <p>
 * For example, with the following class definitions:
 *
 * <pre>
 * class Animal {}
 * class Dog extends Animal {}
 * class Cat extends Animal {}
 * </pre>
 *
 * <p>
 * The user would be required to create {@link JAXBContext} as
 * <tt>JAXBContext.newInstance(Dog.class,Cat.class)</tt>
 * (<tt>Animal</tt> will be automatically picked up since <tt>Dog</tt>
 * and <tt>Cat</tt> refers to it.)
 *
 * <p>
 * {@link XmlSeeAlso} annotation would allow you to write:
 * <pre>
 * &#64;XmlSeeAlso({Dog.class,Cat.class})
 * class Animal {}
```

```
 * class Dog extends Animal {}
 * class Cat extends Animal {}
 * </pre>
 *
 * <p>
 * This would allow you to do <tt>JAXBContext.newInstance(Animal.class)</tt>.
 * By the help of this annotation, JAXB implementations will be able to
 * correctly bind <tt>Dog</tt> and <tt>Cat</tt>.
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB2.1
 */
@Target({ElementType.TYPE})
@Retention(RUNTIME)
public @interface XmlSeeAlso {
    Class[] value();
}
```

3.33 XmlTransient.java

Secuencia 67: javax/xml/bind/annotation/XmlTransient.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

```

```
/**
 * <p>
 * Prevents the mapping of a JavaBean property/type to XML representation.
 * <p>
 * The <tt>@XmlTransient</tt> annotation is useful for resolving name
 * collisions between a JavaBean property name and a field name or
 * preventing the mapping of a field/property. A name collision can
 * occur when the decapitalized JavaBean property name and a field
 * name are the same. If the JavaBean property refers to the field,
 * then the name collision can be resolved by preventing the
 * mapping of either the field or the JavaBean property using the
 * <tt>@XmlTransient</tt> annotation.
 *
 * <p>
 * When placed on a class, it indicates that the class shouldn't be mapped
 * to XML by itself. Properties on such class will be mapped to XML along
 * with its derived classes, as if the class is inlined.
 *
 * <p><b>Usage</b></p>
 * <p> The <tt>@XmlTransient</tt> annotation can be used with the following
 *     program elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> field </li>
 *   <li> class </li>
 * </ul>
 *
 * <p><tt>@XmlTransient</tt>is mutually exclusive with all other
 * JAXB defined annotations. </p>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * <p><b>Example:</b> Resolve name collision between JavaBean property and
 *     field name </p>
 *
 * <pre>
 *   // Example: Code fragment
 *   public class USAddress {
 *
 *       // The field name "name" collides with the property name
 *       // obtained by bean decapitalization of getName() below
 *       &#64;XmlTransient public String name;
 *
 *       String getName() {..};
 *       String setName() {..};
 *   }
 *
 *
 *   &lt;!-- Example: XML Schema fragment -->
 *   &lt;xs:complexType name="USAddress">
 *     &lt;xs:sequence>
 *       &lt;xs:element name="name" type="xs:string"/>
```

```
 *     &lt;/xs:sequence>
 *   &lt;/xs:complexType>
 * </pre>
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD, TYPE})
public @interface XmlTransient {}
```

3.34 XmlType.java

Secuencia 68: javax/xml/bind/annotation/XmlType.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import static java.lang.annotation.ElementType.TYPE;
import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * <p>
 * Maps a class or an enum type to a XML Schema type.
 *
 * <p><b>Usage</b></p>
 * <p> The <tt>@XmlType</tt> annnotation can be used with the following program
 * elements:
```

```
* <ul>
*   <li> a top level class </li>
*   <li> an enum type </li>
* </ul>
*
* <p>See "Package Specification" in javax.xml.bind.package javadoc for
* additional common information.</p>
*
* <h3> Mapping a Class </h3>
* <p>
* A class maps to a XML Schema type. A class is a data container for
* values represented by properties and fields. A schema type is a
* data container for values represented by schema components within a
* schema type's content model (e.g. model groups, attributes etc).
* <p> To be mapped, a class must either have a public no-arg
* constructor or a static no-arg factory method. The static factory
* method can be specified in <tt>factoryMethod()</tt> and
* <tt>factoryClass()</tt> annotation elements. The static factory
* method or the no-arg constructor is used during unmarshalling to
* create an instance of this class. If both are present, the static
* factory method overrides the no-arg constructor.
* <p>
* A class maps to either a XML Schema complex type or a XML Schema simple
* type. The XML Schema type is derived based on the
* mapping of JavaBean properties and fields contained within the
* class. The schema type to which the class is mapped can either be
* named or anonymous. A class can be mapped to an anonymous schema
* type by annotating the class with <tt>&#64XmlType(name="")</tt>.
* <p>
* Either a global element, local element or a local attribute can be
* associated with an anonymous type as follows:
* <ul>
*   <li><b>global element: </b> A global element of an anonymous
*      type can be derived by annotating the class with @{@link
*      XmlRootElement}. See Example 3 below. </li>
*
*   <li><b>local element: </b> A JavaBean property that references
*      a class annotated with @XmlType(name="") and is mapped to the
*      element associated with the anonymous type. See Example 4
*      below.</li>
*
*   <li><b>attribute: </b> A JavaBean property that references
*      a class annotated with @XmlType(name="") and is mapped to the
*      attribute associated with the anonymous type. See Example 5 below. </li>
* </ul>
* <b> Mapping to XML Schema Complex Type </b>
* <ul>
*   <li>If class is annotated with <tt>@XmlType(name="") </tt>, it
*   is mapped to an anonymous type otherwise, the class name maps
*   to a complex type name. The <tt>XmlName()</tt> annotation element
*   can be used to customize the name.</li>
*
*   <li> Properties and fields that are mapped to elements are mapped to a
```

```
*   content model within a complex type. The annotation element
*   <tt>propOrder()</tt> can be used to customize the content model to be
*   <tt>xs:all</tt> or <tt>xs:sequence</tt>.  It is used for specifying
*   the order of XML elements in <tt>xs:sequence</tt>. </li>
*
*   <li> Properties and fields can be mapped to attributes within the
*        complex type.  </li>
*
*   <li> The targetnamespace of the XML Schema type can be customized
*        using the annotation element <tt>namespace()</tt>. </li>
* </ul>
*
* <p>
* <b> Mapping class to XML Schema simple type </b>
* <p>
* A class can be mapped to a XML Schema simple type using the
* <tt>@XmlValue</tt> annotation. For additional details and examples,
* see @{@link XmlValue} annotation type.
* <p>
* The following table shows the mapping of the class to a XML Schema
* complex type or simple type. The notational symbols used in the table are:
* <ul>
*   <li> ->    : represents a mapping </li>
*   <li> [x]+  : one or more occurances of x </li>
*   <li> [ <tt>@XmlValue</tt> property ]: JavaBean property annotated with
*         <tt>@XmlValue</tt></li>
*   <li> X     : don't care
* </ul>
* <blockquote>
*   <table border="1" cellpadding="4" cellspacing="3">
*     <tbody>
*       <tr>
*         <td><b>Target</b></td>
*         <td><b>propOrder</b></td>
*         <td><b>ClassBody</b></td>
*         <td><b>ComplexType</b></td>
*         <td><b>SimpleType</b></td>
*       </tr>
*
*       <tr valign="top">
*         <td>Class</td>
*         <td>{}</td>
*         <td>[property]+ -> elements</td>
*         <td>complexcontent<br>xs:all</td>
*         <td> </td>
*       </tr>
*
*       <tr valign="top">
*         <td>Class</td>
*         <td>non empty</td>
*         <td>[property]+ -> elements</td>
*         <td>complexcontent<br>xs:sequence</td>
*         <td> </td>
```

```
*       </tr>
*
*       <tr valign="top">
*         <td>Class</td>
*         <td>X</td>
*         <td>no property -> element</td>
*         <td>complexcontent<br>empty sequence</td>
*         <td> </td>
*       </tr>
*
*       <tr valign="top">
*         <td>Class</td>
*         <td>X</td>
*         <td>1 [ <tt>@XmlValue</tt> property] && <br> [property]+
*             ->attributes</td>
*         <td>simplecontent</td>
*         <td> </td>
*       </tr>
*
*       <tr valign="top">
*         <td>Class</td>
*         <td>X</td>
*         <td>1 [ <tt>@XmlValue</tt> property ]&& <br> no properties
*         -> attribute</td>
*         <td> </td>
*         <td>simpletype</td>
*         <td> </td>
*       </tr>
*     </tbody>
*   </table>
* </blockquote>
*
* <h3> Mapping an enum type </h3>
*
* An enum type maps to a XML schema simple type with enumeration
* facets. The following annotation elements are ignored since they
* are not meaningful: <tt>propOrder()</tt> , <tt>factoryMethod()</tt> ,
* <tt>factoryClass()</tt> .
*
*  <h3> Usage with other annotations </h3>
* <p> This annotation can be used with the following annotations:
* {@link XmlRootElement}, {@link XmlAccessorOrder}, {@link XmlAccessorType},
* {@link XmlEnum}. However, {@link
* XmlAccessorOrder} and {@link XmlAccessorType} are ignored when this
* annotation is used on an enum type.
*
* <p> <b> Example 1: </b> Map a class to a complex type with
*   xs:sequence with a customized ordering of JavaBean properties.
* </p>
*
* <pre>
*   &#64;XmlType(propOrder={"street", "city" , "state", "zip", "name" })
*   public class USAddress {
```

```
*     String getName() {..};
*     void setName(String) {..};
*
*     String getStreet() {..};
*     void setStreet(String) {..};
*
*     String getCity() {..};
*     void setCity(String) {..};
*
*     String getState() {..};
*     void setState(String) {..};
*
*     java.math.BigDecimal getZip() {..};
*     void setZip(java.math.BigDecimal) {..};
*   }
*
*   &lt;!-- XML Schema mapping for USAddress -->
*   &lt;xs:complexType name="USAddress">
*     &lt;xs:sequence>
*       &lt;xs:element name="street" type="xs:string"/>
*       &lt;xs:element name="city" type="xs:string"/>
*       &lt;xs:element name="state" type="xs:string"/>
*       &lt;xs:element name="zip" type="xs:decimal"/>
*       &lt;xs:element name="name" type="xs:string"/>
*     &lt;/xs:all>
*   &lt;/xs:complexType>
* </pre>
* <p> <b> Example 2: </b> Map a class to a complex type with
*     xs:all </p>
* <pre>
* &#64;XmlType(propOrder={})
* public class USAddress { ...}
*
* &lt;!-- XML Schema mapping for USAddress -->
* &lt;xs:complexType name="USAddress">
*   &lt;xs:all>
*     &lt;xs:element name="name" type="xs:string"/>
*     &lt;xs:element name="street" type="xs:string"/>
*     &lt;xs:element name="city" type="xs:string"/>
*     &lt;xs:element name="state" type="xs:string"/>
*     &lt;xs:element name="zip" type="xs:decimal"/>
*   &lt;/xs:sequence>
* &lt;/xs:complexType>
*</pre>
* <p> <b> Example 3: </b> Map a class to a global element with an
* anonymous type.
* </p>
* <pre>
*   &#64;XmlRootElement
*   &#64;XmlType(name="")
*   public class USAddress { ...}
*
*   &lt;!-- XML Schema mapping for USAddress -->
```

```
*   &lt;xs:element name="USAddress">
*     &lt;xs:complexType>
*       &lt;xs:sequence>
*         &lt;xs:element name="name" type="xs:string"/>
*         &lt;xs:element name="street" type="xs:string"/>
*         &lt;xs:element name="city" type="xs:string"/>
*         &lt;xs:element name="state" type="xs:string"/>
*         &lt;xs:element name="zip" type="xs:decimal"/>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
*   &lt;/xs:element>
* </pre>
*
* <p> <b> Example 4: </b> Map a property to a local element with
* anonmyous type.
* <pre>
*   //Example: Code fragment
*   public class Invoice {
*       USAddress addr;
*           ...
*       }
*
*   &#64;XmlType(name="")
*   public class USAddress { ... }
*   }
*
*   &lt;!-- XML Schema mapping for USAddress -->
*   &lt;xs:complexType name="Invoice">
*     &lt;xs:sequence>
*       &lt;xs:element name="addr">
*         &lt;xs:complexType>
*           &lt;xs:element name="name", type="xs:string"/>
*           &lt;xs:element name="city", type="xs:string"/>
*           &lt;xs:element name="city" type="xs:string"/>
*           &lt;xs:element name="state" type="xs:string"/>
*           &lt;xs:element name="zip" type="xs:decimal"/>
*         &lt;/xs:complexType>
*       ...
*     &lt;/xs:sequence>
*   &lt;/xs:complexType>
* </pre>
*
* <p> <b> Example 5: </b> Map a property to an attribute with
* anonymous type.
*
* <pre>
*
*     //Example: Code fragment
*     public class Item {
*         public String name;
*         &#64;XmlAttribute
*         public USPrice price;
*     }
```

```
*
*     // map class to anonymous simple type.
*     &#64;XmlType(name="")
*     public class USPrice {
*         &#64;XmlValue
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example: XML Schema fragment -->
*     &lt;xs:complexType name="Item">
*       &lt;xs:sequence>
*         &lt;xs:element name="name" type="xs:string"/>
*         &lt;xs:attribute name="price">
*           &lt;xs:simpleType>
*             &lt;xs:restriction base="xs:decimal"/>
*           &lt;/xs:simpleType>
*         &lt;/xs:attribute>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
* </pre>
*
*  <p> <b> Example 6: </b> Define a factoryClass and factoryMethod
*
* <pre>
*      &#64;XmlType(name="USAddressType", factoryClass=USAddressFactory.class,
*      factoryMethod="getUSAddress")
*      public class USAddress {
*
*          private String city;
*          private String name;
*          private String state;
*          private String street;
*          private int    zip;
*
*      public USAddress(String name, String street, String city,
*          String state, int zip) {
*          this.name = name;
*          this.street = street;
*          this.city = city;
*          this.state = state;
*          this.zip = zip;
*      }
*  }
*
*  public class USAddressFactory {
*      public static USAddress getUSAddress(){
*       return new USAddress("Mark Baker", "23 Elm St",
*          "Dayton", "OH", 90952);
*  }
*
* </pre>
*
*  <p> <b> Example 7: </b> Define factoryMethod and use the default factoryClass
```

```
 *
 * <pre>
 *      &#64;XmlType(name="USAddressType", factoryMethod="getNewInstance")
 *      public class USAddress {
 *
 *          private String city;
 *          private String name;
 *          private String state;
 *          private String street;
 *          private int    zip;
 *
 *          private USAddress() {}
 *
 *          public static USAddress getNewInstance(){
 *              return new USAddress();
 *          }
 *      }
 * </pre>
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @see XmlElement
 * @see XmlAttribute
 * @see XmlValue
 * @see XmlSchema
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({TYPE})
public @interface XmlType {
    /**
     * Name of the XML Schema type which the class is mapped.
     */
    String name() default "##default" ;

    /**
     * Specifies the order for XML Schema elements when class is
     * mapped to a XML Schema complex type.
     *
     * <p> Refer to the table for how the propOrder affects the
     * mapping of class </p>
     *
     * <p> The propOrder is a list of names of JavaBean properties in
     *     the class. Each name in the list is the name of a Java
     *     identifier of the JavaBean property. The order in which
     *     JavaBean properties are listed is the order of XML Schema
     *     elements to which the JavaBean properties are mapped. </p>
     * <p> All of the JavaBean properties being mapped to XML Schema elements
     *     must be listed.
     * <p> A JavaBean property or field listed in propOrder must not
     *     be transient or annotated with <tt>@XmlTransient</tt>.
     * <p> The default ordering of JavaBean properties is determined
     *     by @{@link XmlAccessorOrder}.
     */
```

```
    String[] propOrder() default {""};

    /**
     * Name of the target namespace of the XML Schema type. By
     * default, this is the target namespace to which the package
     * containing the class is mapped.
     */
    String namespace() default "##default" ;

    /**
     * Class containing a no-arg factory method for creating an
     * instance of this class. The default is this class.
     *
     * <p>If <tt>factoryClass</tt> is DEFAULT.class and
     * <tt>factoryMethod</tt> is "", then there is no static factory
     * method.
     *
     * <p>If <tt>factoryClass</tt> is DEFAULT.class and
     * <tt>factoryMethod</tt> is not "", then
     * <tt>factoryMethod</tt> is the name of a static factory method
     * in this class.
     *
     * <p>If <tt>factoryClass</tt> is not DEFAULT.class, then
     * <tt>factoryMethod</tt> must not be "" and must be the name of
     * a static factory method specified in <tt>factoryClass</tt>.
     */
    Class factoryClass() default DEFAULT.class;

    /**
     * Used in {@link XmlType#factoryClass()} to
     * signal that either factory mehod is not used or
     * that it's in the class with this {@link XmlType} itself.
     */
    static final class DEFAULT {}

    /**
     * Name of a no-arg factory method in the class specified in
     * <tt>factoryClass</tt> factoryClass().
     *
     */
    String factoryMethod() default "";
}
```

3.35 XmlValue.java

Secuencia 69: javax/xml/bind/annotation/XmlValue.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation;

import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * <p>
 * Enables mapping a class to a  XML Schema complex type with a
 * simpleContent or a XML Schema simple type.
 * </p>
 *
 * <p>
 * <b> Usage: </b>
 * <p>
 * The <tt>@XmlValue</tt> annotation can be used with the following program
 * elements:
 * <ul>
 *   <li> a JavaBean property.</li>
 *   <li> non static, non transient field.</li>
 * </ul>
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * The usage is subject to the following usage constraints:
 * <ul>
 *   <li>At most one field or property can be annotated with the
 *       <tt>@XmlValue</tt> annotation. </li>
 *
 *   <li><tt>@XmlValue</tt> can be used with the following
 *   annotations: {@link XmlList}. However this is redundant since
 *   {@link XmlList} maps a type to a simple schema type that derives by
 *   list just as {@link XmlValue} would. </li>
```

```
*
*   <li>If the type of the field or property is a collection type,
*       then the collection item type must map to a simple schema
*       type.  </li>
*
*   <li>If the type of the field or property is not a collection
*       type, then the type must map to a XML Schema simple type. </li>
*
* </ul>
* </p>
* <p>
* If the annotated JavaBean property is the sole class member being
* mapped to XML Schema construct, then the class is mapped to a
* simple type.
*
* If there are additional JavaBean properties (other than the
* JavaBean property annotated with <tt>@XmlValue</tt> annotation)
* that are mapped to XML attributes, then the class is mapped to a
* complex type with simpleContent.
* </p>
*
* <p> <b> Example 1: </b> Map a class to XML Schema simpleType</p>
*
*   <pre>
*
*     // Example 1: Code fragment
*     public class USPrice {
*         &#64;XmlValue
*         public java.math.BigDecimal price;
*     }
*
*     &lt;!-- Example 1: XML Schema fragment -->
*     &lt;xs:simpleType name="USPrice">
*       &lt;xs:restriction base="xs:decimal"/>
*     &lt;/xs:simpleType>
*
*   </pre>
*
* <p><b> Example 2: </b> Map a class to XML Schema complexType with
*        with simpleContent.</p>
*
*   <pre>
*
*   // Example 2: Code fragment
*   public class InternationalPrice {
*       &#64;XmlValue
*       public java.math.BigDecimal price;
*
*       &#64;XmlAttribute
*       public String currency;
*   }
*
*   &lt;!-- Example 2: XML Schema fragment -->
```

```
 *   &lt;xs:complexType name="InternationalPrice">
 *     &lt;xs:simpleContent>
 *       &lt;xs:extension base="xs:decimal">
 *         &lt;xs:attribute name="currency" type="xs:string"/>
 *       &lt;/xs:extension>
 *     &lt;/xs:simpleContent>
 *   &lt;/xs:complexType>
 *
 *   </pre>
 * </p>
 *
 * @author Sekhar Vajjhala, Sun Microsystems, Inc.
 * @see XmlType
 * @since JAXB2.0
 */

@Retention(RUNTIME) @Target({FIELD, METHOD})
public @interface XmlValue {}
```

4 Xml Bind Annotation Adapters (javax.xml.bind.annotation.adapters package)

4.1 CollapsedStringAdapter.java

Secuencia 70: javax/xml/bind/annotation/adapters/CollapsedStringAdapter.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;

```

```java


/**
 * Built-in {@link XmlAdapter} to handle <tt>xs:token</tt> and its derived types.
 *
 * <p>
 * This adapter removes leading and trailing whitespaces, then truncate any
 * sequnce of tab, CR, LF, and SP by a single whitespace character ' '.
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB 2.0
 */
public class CollapsedStringAdapter extends XmlAdapter<String,String> {
    /**
     * Removes leading and trailing whitespaces of the string
     * given as the parameter, then truncate any
     * sequnce of tab, CR, LF, and SP by a single whitespace character ' '.
     */
    public String unmarshal(String text) {
        if(text==null)  return null;        // be defensive

        int len = text.length();

        // most of the texts are already in the collapsed form.
        // so look for the first whitespace in the hope that we will
        // never see it.
        int s=0;
        while(s<len) {
            if(isWhiteSpace(text.charAt(s)))
                break;
            s++;
        }
        if(s==len)
            // the input happens to be already collapsed.
            return text;

        // we now know that the input contains spaces.
        // let's sit down and do the collapsing normally.

        StringBuilder result = new StringBuilder(len /*allocate enough size to avoid re-allocation
            */ );

        if(s!=0) {
            for( int i=0; i<s; i++ )
                result.append(text.charAt(i));
            result.append(' ');
        }

        boolean inStripMode = true;
        for (int i = s+1; i < len; i++) {
            char ch = text.charAt(i);
            boolean b = isWhiteSpace(ch);
            if (inStripMode && b)
```

```
                continue; // skip this character

            inStripMode = b;
            if (inStripMode)
                result.append(' ');
            else
                result.append(ch);
        }

        // remove trailing whitespaces
        len = result.length();
        if (len > 0 && result.charAt(len - 1) == ' ')
            result.setLength(len - 1);
        // whitespaces are already collapsed,
        // so all we have to do is to remove the last one character
        // if it's a whitespace.

        return result.toString();
    }

    /**
     * No-op.
     *
     * Just return the same string given as the parameter.
     */
    public String marshal(String s) {
        return s;
    }


    /** returns true if the specified char is a white space character. */
    protected static boolean isWhiteSpace(char ch) {
        // most of the characters are non-control characters.
        // so check that first to quickly return false for most of the cases.
        if( ch>0x20 )   return false;

        // other than we have to do four comparisons.
        return ch == 0x9 || ch == 0xA || ch == 0xD || ch == 0x20;
    }
}
```

4.2 HexBinaryAdapter.java

Secuencia 71: javax/xml/bind/annotation/adapters/HexBinaryAdapter.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;

import javax.xml.bind.DatatypeConverter;

/**
 * {@link XmlAdapter} for <tt>xs:hexBinary</tt>.
 *
 * <p>
 * This {@link XmlAdapter} binds <tt>byte[]</tt> to the hexBinary representation in XML.
 *
 * @author Kohsuke Kawaguchi
 * @since JAXB 2.0
 */
public final class HexBinaryAdapter extends XmlAdapter<String,byte[]> {
    public byte[] unmarshal(String s) {
        if(s==null)     return null;
        return DatatypeConverter.parseHexBinary(s);
    }

    public String marshal(byte[] bytes) {
        if(bytes==null)     return null;
        return DatatypeConverter.printHexBinary(bytes);
    }
}
```

4.3 NormalizedStringAdapter.java

Secuencia 72: javax/xml/bind/annotation/adapters/NormalizedStringAdapter.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;



/**
 * {@link XmlAdapter} to handle <tt>xs:normalizedString</tt>.
 *
 * <p>
 * Replaces any tab, CR, and LF by a whitespace character ' ',
 * as specified in <a href="http://www.w3.org/TR/xmlschema-2/#rf-whiteSpace">the whitespace facet '
     replace'</a>
 *
 * @author Kohsuke Kawaguchi, Martin Grebac
 * @since JAXB 2.0
 */
public final class NormalizedStringAdapter extends XmlAdapter<String,String> {
    /**
     * Replace any tab, CR, and LF by a whitespace character ' ',
     * as specified in <a href="http://www.w3.org/TR/xmlschema-2/#rf-whiteSpace">the whitespace
         facet 'replace'</a>
     */
    public String unmarshal(String text) {
        if(text==null)      return null;    // be defensive

        int i=text.length()-1;

        // look for the first whitespace char.
        while( i>=0 && !isWhiteSpaceExceptSpace(text.charAt(i)) )
            i--;

        if( i<0 )
            // no such whitespace. replace(text)==text.
            return text;

        // we now know that we need to modify the text.
        // allocate a char array to do it.
        char[] buf = text.toCharArray();
```

```java

        buf[i--] = ' ';
        for( ; i>=0; i-- )
            if( isWhiteSpaceExceptSpace(buf[i]))
                buf[i] = ' ';

        return new String(buf);
    }

    /**
     * No-op.
     *
     * Just return the same string given as the parameter.
     */
        public String marshal(String s) {
            return s;
        }


    /**
     * Returns true if the specified char is a white space character
     * but not 0x20.
     */
    protected static boolean isWhiteSpaceExceptSpace(char ch) {
        // most of the characters are non-control characters.
        // so check that first to quickly return false for most of the cases.
        if( ch>=0x20 )   return false;

        // other than we have to do four comparisons.
        return ch == 0x9 || ch == 0xA || ch == 0xD;
    }
}
```

4.4 XmlAdapter.java

Secuencia 73: javax/xml/bind/annotation/adapters/XmlAdapter.java

```java
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;

/**
 * Adapts a Java type for custom marshaling.
 *
 * <p> <b> Usage: </b> </p>
 *
 * <p>
 * Some Java types do not map naturally to a XML representation, for
 * example <tt>HashMap</tt> or other non JavaBean classes. Conversely,
 * a XML repsentation may map to a Java type but an application may
 * choose to accesss the XML representation using another Java
 * type. For example, the schema to Java binding rules bind
 * xs:DateTime by default to XmlGregorianCalendar. But an application
 * may desire to bind xs:DateTime to a custom type,
 * MyXmlGregorianCalendar, for example. In both cases, there is a
 * mismatch between <i> bound type </i>, used by an application to
 * access XML content and the <i> value type</i>, that is mapped to an
 * XML representation.
 *
 * <p>
 * This abstract class defines methods for adapting a bound type to a value
 * type or vice versa. The methods are invoked by the JAXB binding
 * framework during marshaling and unmarshalling:
 *
 * <ul>
 *   <li> <b> XmlAdapter.marshal(...): </b> During marshalling, JAXB
 *        binding framework invokes XmlAdapter.marshal(..) to adapt a
 *        bound type to value type, which is then marshaled to XML
 *        representation. </li>
 *
 *   <li> <b> XmlAdapter.unmarshal(...): </b> During unmarshalling,
 *        JAXB binding framework first unmarshals XML representation
 *        to a value type and then invokes XmlAdapter.unmarshal(..) to
 *        adapt the value type to a bound type. </li>
 * </ul>
 *
 * Writing an adapter therefore involves the following steps:
 *
 * <ul>
 *   <li> Write an adapter that implements this abstract class. </li>
 *   <li> Install the adapter using the annotation {@link
 *        XmlJavaTypeAdapter} </li>
 * </ul>
 *
```

```
* <p><b>Example:</b> Customized mapping of <tt>HashMap</tt></p>
* <p> The following example illustrates the use of
* <tt>&#64;XmlAdapter</tt> and <tt>&#64;XmlJavaTypeAdapter</tt> to
* customize the mapping of a <tt>HashMap</tt>.
*
* <p> <b> Step 1: </b> Determine the desired XML representation for HashMap.
*
* <pre>
*     &lt;hashmap>
*         &lt;entry key="id123">this is a value&lt;/entry>
*         &lt;entry key="id312">this is another value&lt;/entry>
*         ...
*       &lt;/hashmap>
* </pre>
*
* <p> <b> Step 2: </b> Determine the schema definition that the
* desired XML representation shown above should follow.
*
* <pre>
*
*     &lt;xs:complexType name="myHashMapType">
*       &lt;xs:sequence>
*         &lt;xs:element name="entry" type="myHashMapEntryType"
*                         minOccurs = "0" maxOccurs="unbounded"/>
*       &lt;/xs:sequence>
*     &lt;/xs:complexType>
*
*     &lt;xs:complexType name="myHashMapEntryType">
*       &lt;xs:simpleContent>
*         &lt;xs:extension base="xs:string">
*           &lt;xs:attribute name="key" type="xs:int"/>
*         &lt;/xs:extension>
*       &lt;/xs:simpleContent>
*     &lt;/xs:complexType>
*
* </pre>
*
* <p> <b> Step 3: </b> Write value types that can generate the above
* schema definition.
*
* <pre>
*     public class MyHashMapType {
*         List&lt;MyHashMapEntryType> entry;
*     }
*
*     public class MyHashMapEntryType {
*         &#64;XmlAttribute
*         public Integer key;
*
*         &#64;XmlValue
*         public String value;
*     }
* </pre>
```

```
 *
 * <p> <b> Step 4: </b> Write the adapter that adapts the value type,
 * MyHashMapType to a bound type, HashMap, used by the application.
 *
 * <pre>
 *     public final class MyHashMapAdapter extends
 *                        XmlAdapter&lt;MyHashMapType,HashMap> { ... }
 *
 * </pre>
 *
 * <p> <b> Step 5: </b> Use the adapter.
 *
 * <pre>
 *     public class Foo {
 *         &#64;XmlJavaTypeAdapter(MyHashMapAdapter.class)
 *         HashMap hashmap;
 *         ...
 *     }
 * </pre>
 *
 * The above code fragment will map to the following schema:
 *
 * <pre>
 *     &lt;xs:complexType name="Foo">
 *       &lt;xs:sequence>
 *         &lt;xs:element name="hashmap" type="myHashMapType"
 *       &lt;/xs:sequence>
 *     &lt;/xs:complexType>
 * </pre>
 *
 * @param <BoundType>
 *      The type that JAXB doesn't know how to handle. An adapter is written
 *      to allow this type to be used as an in-memory representation through
 *      the <tt>ValueType</tt>.
 * @param <ValueType>
 *      The type that JAXB knows how to handle out of the box.
 *
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems Inc.</li> <li> Kohsuke Kawaguchi, Sun
     Microsystems Inc.</li></ul>
 * @see XmlJavaTypeAdapter
 * @since JAXB 2.0
 */
public abstract class XmlAdapter<ValueType,BoundType> {

    /**
     * Do-nothing constructor for the derived classes.
     */
    protected XmlAdapter() {}

    /**
     * Convert a value type to a bound type.
     *
     * @param v
```

```
     *      The value to be converted. Can be null.
     * @throws Exception
     *      if there's an error during the conversion. The caller is responsible for
     *      reporting the error to the user through {@link javax.xml.bind.ValidationEventHandler}.
     */
    public abstract BoundType unmarshal(ValueType v) throws Exception;

    /**
     * Convert a bound type to a value type.
     *
     * @param v
     *      The value to be convereted. Can be null.
     * @throws Exception
     *      if there's an error during the conversion. The caller is responsible for
     *      reporting the error to the user through {@link javax.xml.bind.ValidationEventHandler}.
     */
    public abstract ValueType marshal(BoundType v) throws Exception;
}
```

4.5 XmlJavaTypeAdapter.java

Secuencia 74: javax/xml/bind/annotation/adapters/XmlJavaTypeAdapter.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;

import javax.xml.bind.annotation.XmlAnyElement;
import javax.xml.bind.annotation.XmlElementRefs;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlSchemaType;
```

```java
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlSchema;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlSchemaTypes;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;

import static java.lang.annotation.RetentionPolicy.RUNTIME;
import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.ElementType.PACKAGE;


/**
 * Use an adapter that implements {@link XmlAdapter} for custom marshaling.
 *
 * <p> <b> Usage: </b> </p>
 *
 * <p> The <tt>@XmlJavaTypeAdapter</tt> annotation can be used with the
 * following program elements:
 * <ul>
 *   <li> a JavaBean property </li>
 *   <li> field </li>
 *   <li> parameter </li>
 *   <li> package </li>
 *   <li> from within {@link XmlJavaTypeAdapters} </li>
 * </ul>
 *
 * <p> When <tt>@XmlJavaTypeAdapter</tt> annotation is defined on a
 * class, it applies to all references to the class.
 * <p> When <tt>@XmlJavaTypeAdapter</tt> annotation is defined at the
 * package level it applies to all references from within the package
 * to <tt>@XmlJavaTypeAdapter.type()</tt>.
 * <p> When <tt>@XmlJavaTypeAdapter</tt> annotation is defined on the
 * field, property or parameter, then the annotation applies to the
 * field, property or the parameter only.
 * <p> A <tt>@XmlJavaTypeAdapter</tt> annotation on a field, property
 * or parameter overrides the <tt>@XmlJavaTypeAdapter</tt> annotation
 * associated with the class being referenced by the field, property
 * or parameter.
 * <p> A <tt>@XmlJavaTypeAdapter</tt> annotation on a class overrides
 * the <tt>@XmlJavaTypeAdapter</tt> annotation specified at the
 * package level for that class.
 *
 * <p>This annotation can be used with the following other annotations:
 * {@link XmlElement}, {@link XmlAttribute}, {@link XmlElementRef},
 * {@link XmlElementRefs}, {@link XmlAnyElement}. This can also be
 * used at the package level with the following annotations:
 * {@link XmlAccessorType}, {@link XmlSchema}, {@link XmlSchemaType},
 * {@link XmlSchemaTypes}.
```

```
 *
 * <p><b> Example: </b> See example in {@link XmlAdapter}
 *
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems Inc.</li> <li> Kohsuke Kawaguchi, Sun
     Microsystems Inc.</li></ul>
 * @since JAXB2.0
 * @see XmlAdapter
 */

@Retention(RUNTIME) @Target({PACKAGE,FIELD,METHOD,TYPE,PARAMETER})
public @interface XmlJavaTypeAdapter {
    /**
     * Points to the class that converts a value type to a bound type or vice versa.
     * See {@link XmlAdapter} for more details.
     */
    Class<? extends XmlAdapter> value();

    /**
     * If this annotation is used at the package level, then value of
     * the type() must be specified.
     */

    Class type() default DEFAULT.class;

    /**
     * Used in {@link XmlJavaTypeAdapter#type()} to
     * signal that the type be inferred from the signature
     * of the field, property, parameter or the class.
     */

    static final class DEFAULT {}

}
```

4.6 XmlJavaTypeAdapters.java

Secuencia 75: javax/xml/bind/annotation/adapters/XmlJavaTypeAdapters.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.annotation.adapters;

import static java.lang.annotation.ElementType.PACKAGE;
import java.lang.annotation.Retention;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Target;

/**
 * <p>
 * A container for multiple @{@link XmlJavaTypeAdapter} annotations.
 *
 * <p> Multiple annotations of the same type are not allowed on a program
 * element. This annotation therefore serves as a container annotation
 * for multiple &#64;XmlJavaTypeAdapter as follows:
 *
 * <pre>
 * &#64;XmlJavaTypeAdapters ({ @XmlJavaTypeAdapter(...),@XmlJavaTypeAdapter(...) })
 * </pre>
 *
 * <p>The <tt>@XmlJavaTypeAdapters</tt> annnotation is useful for
 * defining {@link XmlJavaTypeAdapter} annotations for different types
 * at the package level.
 *
 * <p>See "Package Specification" in javax.xml.bind.package javadoc for
 * additional common information.</p>
 *
 * @author <ul><li>Sekhar Vajjhala, Sun Microsystems, Inc.</li></ul>
 * @see XmlJavaTypeAdapter
 * @since JAXB2.0
 */
@Retention(RUNTIME) @Target({PACKAGE})
public @interface XmlJavaTypeAdapters {
    /**
     * Collection of @{@link XmlJavaTypeAdapter} annotations
     */
    XmlJavaTypeAdapter[] value();
}
```

5 Xml Bind Attachment (javax.xml.bind.attachment package)

5.1 AttachmentMarshaller.java

Secuencia 76: javax/xml/bind/attachment/AttachmentMarshaller.java

```
/*
```

```
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.attachment;

import javax.activation.DataHandler;
import javax.xml.bind.Marshaller;

/**
 * <p>Enable JAXB marshalling to optimize storage of binary data.</p>
 *
 * <p>This API enables an efficient cooperative creation of optimized
 * binary data formats between a JAXB marshalling process and a MIME-based package
 * processor. A JAXB implementation marshals the root body of a MIME-based package,
 * delegating the creation of referenceable MIME parts to
 * the MIME-based package processor that implements this abstraction.</p>
 *
 * <p>XOP processing is enabled when {@link #isXOPPackage()} is true.
 *    See {@link #addMtomAttachment(DataHandler, String, String)} for details.
 * </p>
 *
 * <p>WS-I Attachment Profile 1.0 is supported by
 * {@link #addSwaRefAttachment(DataHandler)} being called by the
 * marshaller for each JAXB property related to
 * {http://ws-i.org/profiles/basic/1.1/xsd}swaRef.</p>
 *
 *
 * @author Marc Hadley
 * @author Kohsuke Kawaguchi
 * @author Joseph Fialli
 * @since JAXB 2.0
 *
```

```
 * @see Marshaller#setAttachmentMarshaller(AttachmentMarshaller)
 *
 * @see <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/">XML-binary Optimized Packaging</a>
 * @see <a href="http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html">WS-I
     Attachments Profile Version 1.0.</a>
 */
public abstract class AttachmentMarshaller {

    /**
     * <p>Consider MIME content <code>data</code> for optimized binary storage as an attachment.
     *
     * <p>
     * This method is called by JAXB marshal process when {@link #isXOPPackage()} is
     * <code>true</code>, for each element whose datatype is "base64Binary", as described in
     * Step 3 in
     * <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/#creating_xop_packages">Creating XOP
         Packages</a>.
     *
     * <p>
     * The method implementor determines whether <code>data</code> shall be attached separately
     * or inlined as base64Binary data. If the implementation chooses to optimize the storage
     * of the binary data as a MIME part, it is responsible for attaching <code>data</code> to the
     * MIME-based package, and then assigning an unique content-id, cid, that identifies
     * the MIME part within the MIME message. This method returns the cid,
     * which enables the JAXB marshaller to marshal a XOP element that refers to that cid in place
     * of marshalling the binary data. When the method returns null, the JAXB marshaller
     * inlines <code>data</code> as base64binary data.
     *
     * <p>
     * The caller of this method is required to meet the following constraint.
     * If the element infoset item containing <code>data</code> has the attribute
     * <code>xmime:contentType</code> or if the JAXB property/field representing
     * <code>data</code>is annotated with a known MIME type,
     * <code>data.getContentType()</code> should be set to that MIME type.
     *
     * <p>
     * The <code>elementNamespace</code> and <code>elementLocalName</code>
     * parameters provide the
     * context that contains the binary data. This information could
     * be used by the MIME-based package processor to determine if the
     * binary data should be inlined or optimized as an attachment.
     *
     * @param data
     *       represents the data to be attached. Must be non-null.
     * @param elementNamespace
     *      the namespace URI of the element that encloses the base64Binary data.
     *      Can be empty but never null.
     * @param elementLocalName
     *      The local name of the element. Always a non-null valid string.
     *
     * @return
     *     a valid content-id URI (see <a href="http://www.w3.org/TR/xop10/#RFC2387">RFC 2387</a>)
         that identifies the attachment containing <code>data</code>.
```

```
     *     Otherwise, null if the attachment was not added and should instead be inlined in the message.
     *
     * @see <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/">XML-binary Optimized Packaging</a>
     * @see <a href="http://www.w3.org/TR/xml-media-types/">Describing Media Content of Binary Data in XML</a>
     */
    public abstract String addMtomAttachment(DataHandler data, String elementNamespace, String elementLocalName);

    /**
     * <p>Consider binary <code>data</code> for optimized binary storage as an attachment.
     *
     * <p>Since content type is not known, the attachment's MIME content type must be set to "application/octet-stream".</p>
     *
     * <p>
     * The <code>elementNamespace</code> and <code>elementLocalName</code>
     * parameters provide the
     * context that contains the binary data. This information could
     * be used by the MIME-based package processor to determine if the
     * binary data should be inlined or optimized as an attachment.
     *
     * @param data
     *      represents the data to be attached. Must be non-null. The actual data region is
     *      specified by <tt>(data,offset,length)</tt> tuple.
     *
     * @param offset
     *       The offset within the array of the first byte to be read;
     *       must be non-negative and no larger than array.length
     *
     * @param length
     *       The number of bytes to be read from the given array;
     *       must be non-negative and no larger than array.length
     *
     * @param mimeType
     *      If the data has an associated MIME type known to JAXB, that is passed
     *      as this parameter. If none is known, "application/octet-stream".
     *      This parameter may never be null.
     *
     * @param elementNamespace
     *      the namespace URI of the element that encloses the base64Binary data.
     *      Can be empty but never null.
     *
     * @param elementLocalName
     *      The local name of the element. Always a non-null valid string.
     *
     * @return content-id URI, cid, to the attachment containing
     *         <code>data</code> or null if data should be inlined.
     *
     * @see #addMtomAttachment(DataHandler, String, String)
     */
```

```java
    public abstract String addMtomAttachment(byte[] data, int offset, int length, String mimeType, String elementNamespace, String elementLocalName);

    /**
     * <p>Read-only property that returns true if JAXB marshaller should enable XOP creation.</p>
     *
     * <p>This value must not change during the marshalling process. When this
     * value is true, the <code>addMtomAttachment(...)</code> method
     * is invoked when the appropriate binary datatypes are encountered by
     * the marshal process.</p>
     *
     * <p>Marshaller.marshal() must throw IllegalStateException if this value is <code>true</code>
     * and the XML content to be marshalled violates Step 1 in
     * <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/#creating_xop_packages">Creating XOP Pacakges</a>
     * http://www.w3.org/TR/2005/REC-xop10-20050125/#creating_xop_packages.
     * <i>"Ensure the Original XML Infoset contains no element information item with a
     * [namespace name] of "http://www.w3.org/2004/08/xop/include" and a [local name] of Include"</i>
     *
     * <p>When this method returns true and during the marshal process
     * at least one call to <code>addMtomAttachment(...)</code> returns
     * a content-id, the MIME-based package processor must label the
     * root part with the application/xop+xml media type as described in
     * Step 5 of
     * <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/#creating_xop_packages">Creating XOP Pacakges</a>.<p>
     *
     * @return true when MIME context is a XOP Package.
     */
    public boolean isXOPPackage() { return false; }

   /**
    * <p>Add MIME <code>data</code> as an attachment and return attachment's content-id, cid.</p>
    *
    * <p>
    * This method is called by JAXB marshal process for each element/attribute typed as
    * {http://ws-i.org/profiles/basic/1.1/xsd}swaRef. The MIME-based package processor
    * implementing this method is responsible for attaching the specified data to a
    * MIME attachment, and generating a content-id, cid, that uniquely identifies the attachment
    * within the MIME-based package.
    *
    * <p>Caller inserts the returned content-id, cid, into the XML content being marshalled.</p>
    *
    * @param data
    *       represents the data to be attached. Must be non-null.
    * @return
    *       must be a valid URI used as cid. Must satisfy Conformance Requirement R2928 from
    *       <a href="http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html#Referencing_Attachments_from_the_SOAP_Envelope">WS-I Attachments Profile Version 1.0.</a>
    */
    public abstract String addSwaRefAttachment(DataHandler data);
}
```

5.2 AttachmentUnmarshaller.java

Secuencia 77: javax/xml/bind/attachment/AttachmentUnmarshaller.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.attachment;

import javax.activation.DataHandler;

/**
 * <p>Enables JAXB unmarshalling of a root document containing optimized binary data formats.</p>
 *
 * <p>This API enables an efficient cooperative processing of optimized
 * binary data formats between a JAXB 2.0 implementation and MIME-based package
 * processor (MTOM/XOP and WS-I AP 1.0). JAXB unmarshals the body of a package, delegating the
 * understanding of the packaging format being used to a MIME-based
 * package processor that implements this abstract class.</p>
 *
 * <p>This abstract class identifies if a package requires XOP processing, {@link #isXOPPackage()}
     and provides retrieval of binary content stored as attachments by content-id.</p>
 *
 * <h2>Identifying the content-id, cid, to pass to <code>getAttachment*(String cid)</code></h2>
 * <ul>
 * <li>
 * For XOP processing, the infoset representation of the cid is described
 * in step 2a in
 * <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/#interpreting_xop_packages">Section 3.2
     Interpreting XOP Packages</a>
 * </li>
 * <li>
```

```
 * For WS-I AP 1.0, the cid is identified as an element or attribute of
 * type <code>ref:swaRef </code> specified in
 * <a href="http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html#
     Referencing_Attachments_from_the_SOAP_Envelope">Section 4.4 Referencing Attachments from the
     SOAP Envelope</a>
 * </li>
 * </ul>
 *
 * @author Marc Hadley
 * @author Kohsuke Kawaguchi
 * @author Joseph Fialli
 *
 * @since JAXB 2.0
 *
 * @see javax.xml.bind.Unmarshaller#setAttachmentUnmarshaller(AttachmentUnmarshaller)
 *
 * @see <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/">XML-binary Optimized Packaging</a>
 * @see <a href="http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html">WS-I
     Attachments Profile Version 1.0.</a>
 * @see <a href="http://www.w3.org/TR/xml-media-types/">Describing Media Content of Binary Data in
     XML</a>
 */
public abstract class AttachmentUnmarshaller {
   /**
    * <p>Lookup MIME content by content-id, <code>cid</code>, and return as a {@link DataHandler
        }.</p>
    *
    * <p>The returned <code>DataHandler</code> instance must be configured
    * to meet the following required mapping constaint.
    * <table border="2" rules="all" cellpadding="4">
    *   <thead>
    *     <tr>
    *       <th align="center" colspan="2">
    *       Required Mappings between MIME and Java Types
    *       </tr>
    *     <tr>
    *       <th>MIME Type</th>
    *       <th>Java Type</th>
    *     </tr>
    *     <tr>
    *       <th><code>DataHandler.getContentType()</code></th>
    *       <th><code>instanceof DataHandler.getContent()</code></th>
    *     </tr>
    *   </thead>
    *   <tbody>
    *     <tr>
    *       <td>image/gif</td>
    *       <td>java.awt.Image</td>
    *     </tr>
    *     <tr>
    *       <td>image/jpeg</td>
    *       <td>java.awt.Image</td>
    *     </tr>
```

```
    *     <tr>
    *       <td>text/xml  or application/xml</td>
    *       <td>javax.xml.transform.Source</td>
    *     </tr>
    *   </tbody>
    *  </table>
    * Note that it is allowable to support additional mappings.</p>
    *
    * @param cid It is expected to be a valid lexical form of the XML Schema
    * <code>xs:anyURI</code> datatype. If <code>{@link #isXOPPackage()}
    * ==true</code>, it must be a valid URI per the <code>cid:</code> URI scheme (see <a href="http://www.ietf.org/rfc/rfc2387.txt">RFC 2387</a>)
    *
    * @return
    *       a {@link DataHandler} that represents the MIME attachment.
    *
    * @throws IllegalArgumentException if the attachment for the given cid is not found.
    */
   public abstract DataHandler getAttachmentAsDataHandler(String cid);

    /**
     * <p>Retrieve the attachment identified by content-id, <code>cid</code>,  as a <tt>byte[]</tt></p>.
     *
     * @param cid It is expected to be a valid lexical form of the XML Schema
     * <code>xs:anyURI</code> datatype. If <code>{@link #isXOPPackage()}
     * ==true</code>, it must be a valid URI per the <code>cid:</code> URI scheme (see <a href="http://www.ietf.org/rfc/rfc2387.txt">RFC 2387</a>)
     *
     * @return byte[] representation of attachment identified by cid.
     *
    * @throws IllegalArgumentException if the attachment for the given cid is not found.
     */
    public abstract byte[] getAttachmentAsByteArray(String cid);

    /**
     * <p>Read-only property that returns true if JAXB unmarshaller needs to perform XOP processing.</p>
     *
     * <p>This method returns <code>true</code> when the constraints specified
     * in  <a href="http://www.w3.org/TR/2005/REC-xop10-20050125/#identifying_xop_documents">Identifying XOP Documents</a> are met.
     * This value must not change during the unmarshalling process.</p>
     *
     * @return true when MIME context is a XOP Document.
     */
    public boolean isXOPPackage() { return false; }
}
```

6 Xml Bind Helpers (javax.xml.bind.helpers package)

6.1 AbstractMarshallerImpl.java

Secuencia 78: javax/xml/bind/helpers/AbstractMarshallerImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.PropertyException;
import javax.xml.bind.ValidationEventHandler;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.attachment.AttachmentMarshaller;
import javax.xml.stream.XMLEventWriter;
import javax.xml.stream.XMLStreamWriter;
import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.sax.SAXResult;
import javax.xml.transform.stream.StreamResult;
import javax.xml.validation.Schema;
import java.io.UnsupportedEncodingException;
import java.io.File;
import java.io.OutputStream;
import java.io.FileOutputStream;
import java.io.BufferedOutputStream;
import java.io.IOException;
// J2SE1.4 feature
// import java.nio.charset.Charset;
// import java.nio.charset.UnsupportedCharsetException;

/**
 * Partial default <tt>Marshaller</tt> implementation.
 *
```

```
 * <p>
 * This class provides a partial default implementation for the
 * {@link javax.xml.bind.Marshaller} interface.
 *
 * <p>
 * The only methods that a JAXB Provider has to implement are
 * {@link Marshaller#marshal(Object, javax.xml.transform.Result) marshal(Object, javax.xml.
     transform.Result)},
 * {@link Marshaller#marshal(Object, javax.xml.transform.Result) marshal(Object, javax.xml.stream.
     XMLStreamWriter)}, and
 * {@link Marshaller#marshal(Object, javax.xml.transform.Result) marshal(Object, javax.xml.stream.
     XMLEventWriter)}.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.Marshaller
 * @since JAXB1.0
 */
public abstract class AbstractMarshallerImpl implements Marshaller
{
    /** handler that will be used to process errors and warnings during marshal */
    private ValidationEventHandler eventHandler =
        new DefaultValidationEventHandler();

    //J2SE1.4 feature
    //private Charset encoding = null;

    /** store the value of the encoding property. */
    private String encoding = "UTF-8";

    /** store the value of the schemaLocation property. */
    private String schemaLocation = null;

    /** store the value of the noNamespaceSchemaLocation property. */
    private String noNSSchemaLocation = null;

    /** store the value of the formattedOutput property. */
    private boolean formattedOutput = false;

    /** store the value of the fragment property. */
    private boolean fragment = false;

    public final void marshal( Object obj, java.io.OutputStream os )
        throws JAXBException {

        checkNotNull( obj, "obj", os, "os" );
        marshal( obj, new StreamResult(os) );
    }

    public void marshal(Object jaxbElement, File output) throws JAXBException {
        checkNotNull(jaxbElement, "jaxbElement", output, "output" );
        try {
            OutputStream os = new BufferedOutputStream(new FileOutputStream(output));
            try {
```

```
                marshal( jaxbElement, new StreamResult(os) );
            } finally {
                os.close();
            }
        } catch (IOException e) {
            throw new JAXBException(e);
        }
    }

    public final void marshal( Object obj, java.io.Writer w )
        throws JAXBException {

        checkNotNull( obj, "obj", w, "writer" );
        marshal( obj, new StreamResult(w) );
    }

    public final void marshal( Object obj, org.xml.sax.ContentHandler handler )
        throws JAXBException {

        checkNotNull( obj, "obj", handler, "handler" );
        marshal( obj, new SAXResult(handler) );
    }

    public final void marshal( Object obj, org.w3c.dom.Node node )
        throws JAXBException {

        checkNotNull( obj, "obj", node, "node" );
        marshal( obj, new DOMResult(node) );
    }

    /**
     * By default, the getNode method is unsupported and throw
     * an {@link java.lang.UnsupportedOperationException}.
     *
     * Implementations that choose to support this method must
     * override this method.
     */
    public org.w3c.dom.Node getNode( Object obj ) throws JAXBException {

        checkNotNull( obj, "obj", Boolean.TRUE, "foo" );

        throw new UnsupportedOperationException();
    }

    /**
     * Convenience method for getting the current output encoding.
     *
     * @return the current encoding or "UTF-8" if it hasn't been set.
     */
    protected String getEncoding() {
        return encoding;
    }

```

```
    /**
     * Convenience method for setting the output encoding.
     *
     * @param encoding a valid encoding as specified in the Marshaller class
     * documentation
     */
    protected void setEncoding( String encoding ) {
        this.encoding = encoding;
    }

    /**
     * Convenience method for getting the current schemaLocation.
     *
     * @return the current schemaLocation or null if it hasn't been set
     */
    protected String getSchemaLocation() {
        return schemaLocation;
    }

    /**
     * Convenience method for setting the schemaLocation.
     *
     * @param location the schemaLocation value
     */
    protected void setSchemaLocation( String location ) {
        schemaLocation = location;
    }

    /**
     * Convenience method for getting the current noNamespaceSchemaLocation.
     *
     * @return the current noNamespaceSchemaLocation or null if it hasn't
     * been set
     */
    protected String getNoNSSchemaLocation() {
        return noNSSchemaLocation;
    }

    /**
     * Convenience method for setting the noNamespaceSchemaLocation.
     *
     * @param location the noNamespaceSchemaLocation value
     */
    protected void setNoNSSchemaLocation( String location ) {
        noNSSchemaLocation = location;
    }

    /**
     * Convenience method for getting the formatted output flag.
     *
     * @return the current value of the formatted output flag or false if
     * it hasn't been set.
     */
```

```
    protected boolean isFormattedOutput() {
        return formattedOutput;
    }

    /**
     * Convenience method for setting the formatted output flag.
     *
     * @param v value of the formatted output flag.
     */
    protected void setFormattedOutput( boolean v ) {
        formattedOutput = v;
    }


    /**
     * Convenience method for getting the fragment flag.
     *
     * @return the current value of the fragment flag or false if
     * it hasn't been set.
     */
    protected boolean isFragment() {
        return fragment;
    }

    /**
     * Convenience method for setting the fragment flag.
     *
     * @param v value of the fragment flag.
     */
    protected void setFragment( boolean v ) {
        fragment = v;
    }


    static String[] aliases = {
        "UTF-8", "UTF8",
        "UTF-16", "Unicode",
        "UTF-16BE", "UnicodeBigUnmarked",
        "UTF-16LE", "UnicodeLittleUnmarked",
        "US-ASCII", "ASCII",
        "TIS-620", "TIS620",

        // taken from the project-X parser
        "ISO-10646-UCS-2", "Unicode",

        "EBCDIC-CP-US", "cp037",
        "EBCDIC-CP-CA", "cp037",
        "EBCDIC-CP-NL", "cp037",
        "EBCDIC-CP-WT", "cp037",

        "EBCDIC-CP-DK", "cp277",
        "EBCDIC-CP-NO", "cp277",
        "EBCDIC-CP-FI", "cp278",
```

```
        "EBCDIC-CP-SE", "cp278",

        "EBCDIC-CP-IT", "cp280",
        "EBCDIC-CP-ES", "cp284",
        "EBCDIC-CP-GB", "cp285",
        "EBCDIC-CP-FR", "cp297",

        "EBCDIC-CP-AR1", "cp420",
        "EBCDIC-CP-HE", "cp424",
        "EBCDIC-CP-BE", "cp500",
        "EBCDIC-CP-CH", "cp500",

        "EBCDIC-CP-ROECE", "cp870",
        "EBCDIC-CP-YU", "cp870",
        "EBCDIC-CP-IS", "cp871",
        "EBCDIC-CP-AR2", "cp918",

        // IANA also defines two that JDK 1.2 doesn't handle:
        //  EBCDIC-CP-GR        --> CP423
        //  EBCDIC-CP-TR        --> CP905
    };

    /**
     * Gets the corresponding Java encoding name from an IANA name.
     *
     * This method is a helper method for the derived class to convert
     * encoding names.
     *
     * @exception UnsupportedEncodingException
     *      If this implementation couldn't find the Java encoding name.
     */
    protected String getJavaEncoding( String encoding ) throws UnsupportedEncodingException {
        try {
            "1".getBytes(encoding);
            return encoding;
        } catch( UnsupportedEncodingException e ) {
            // try known alias
            for( int i=0; i<aliases.length; i+=2 ) {
                if(encoding.equals(aliases[i])) {
                    "1".getBytes(aliases[i+1]);
                    return aliases[i+1];
                }
            }

            throw new UnsupportedEncodingException(encoding);
        }
        /* J2SE1.4 feature
        try {
            this.encoding = Charset.forName( _encoding );
        } catch( UnsupportedCharsetException uce ) {
            throw new JAXBException( uce );
        }
         */
```

```
    }

    /**
     * Default implementation of the setProperty method handles
     * the four defined properties in Marshaller. If a provider
     * needs to handle additional properties, it should override
     * this method in a derived class.
     */
    public void setProperty( String name, Object value )
        throws PropertyException {

        if( name == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "name" ) );
        }

        // recognize and handle four pre-defined properties.
        if( JAXB_ENCODING.equals(name) ) {
            checkString( name, value );
            setEncoding( (String)value );
            return;
        }
        if( JAXB_FORMATTED_OUTPUT.equals(name) ) {
            checkBoolean( name, value );
            setFormattedOutput((Boolean) value );
            return;
        }
        if( JAXB_NO_NAMESPACE_SCHEMA_LOCATION.equals(name) ) {
            checkString( name, value );
            setNoNSSchemaLocation( (String)value );
            return;
        }
        if( JAXB_SCHEMA_LOCATION.equals(name) ) {
            checkString( name, value );
            setSchemaLocation( (String)value );
            return;
        }
        if( JAXB_FRAGMENT.equals(name) )  {
            checkBoolean(name, value);
            setFragment((Boolean) value );
            return;
        }

        throw new PropertyException(name, value);
    }

    /**
     * Default implementation of the getProperty method handles
     * the four defined properties in Marshaller.  If a provider
     * needs to support additional provider specific properties,
     * it should override this method in a derived class.
     */
    public Object getProperty( String name )
```

```
        throws PropertyException {

        if( name == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "name" ) );
        }

        // recognize and handle four pre-defined properties.
        if( JAXB_ENCODING.equals(name) )
            return getEncoding();
        if( JAXB_FORMATTED_OUTPUT.equals(name) )
            return isFormattedOutput()?Boolean.TRUE:Boolean.FALSE;
        if( JAXB_NO_NAMESPACE_SCHEMA_LOCATION.equals(name) )
            return getNoNSSchemaLocation();
        if( JAXB_SCHEMA_LOCATION.equals(name) )
            return getSchemaLocation();
        if( JAXB_FRAGMENT.equals(name) )
            return isFragment()?Boolean.TRUE:Boolean.FALSE;

        throw new PropertyException(name);
    }
    /**
     * @see javax.xml.bind.Marshaller#getEventHandler()
     */
    public ValidationEventHandler getEventHandler() throws JAXBException {
        return eventHandler;
    }

    /**
     * @see javax.xml.bind.Marshaller#setEventHandler(ValidationEventHandler)
     */
    public void setEventHandler(ValidationEventHandler handler)
        throws JAXBException {

        if( handler == null ) {
            eventHandler = new DefaultValidationEventHandler();
        } else {
            eventHandler = handler;
        }
    }




    /*
     * assert that the given object is a Boolean
     */
    private void checkBoolean( String name, Object value ) throws PropertyException {
        if(!(value instanceof Boolean))
            throw new PropertyException(
                Messages.format( Messages.MUST_BE_BOOLEAN, name ) );
    }

```

```
    /*
     * assert that the given object is a String
     */
    private void checkString( String name, Object value ) throws PropertyException {
        if(!(value instanceof String))
            throw new PropertyException(
                Messages.format( Messages.MUST_BE_STRING, name ) );
    }

    /*
     * assert that the parameters are not null
     */
    private void checkNotNull( Object o1, String o1Name,
                               Object o2, String o2Name ) {

        if( o1 == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, o1Name ) );
        }
        if( o2 == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, o2Name ) );
        }
    }

    public void marshal(Object obj, XMLEventWriter writer)
        throws JAXBException {

        throw new UnsupportedOperationException();
    }

    public void marshal(Object obj, XMLStreamWriter writer)
        throws JAXBException {

        throw new UnsupportedOperationException();
    }

    public void setSchema(Schema schema) {
        throw new UnsupportedOperationException();
    }

    public Schema getSchema() {
        throw new UnsupportedOperationException();
    }

    public void setAdapter(XmlAdapter adapter) {
        if(adapter==null)
            throw new IllegalArgumentException();
        setAdapter((Class)adapter.getClass(),adapter);
    }

    public <A extends XmlAdapter> void setAdapter(Class<A> type, A adapter) {
        throw new UnsupportedOperationException();
```

```
    }

    public <A extends XmlAdapter> A getAdapter(Class<A> type) {
        throw new UnsupportedOperationException();
    }

    public void setAttachmentMarshaller(AttachmentMarshaller am) {
        throw new UnsupportedOperationException();
    }

    public AttachmentMarshaller getAttachmentMarshaller() {
        throw new UnsupportedOperationException();
    }

    public void setListener(Listener listener) {
        throw new UnsupportedOperationException();
    }

    public Listener getListener() {
        throw new UnsupportedOperationException();
    }
}
```

6.2 AbstractUnmarshallerImpl.java

Secuencia 79: javax/xml/bind/helpers/AbstractUnmarshallerImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

```

```
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.w3c.dom.Node;

import javax.xml.bind.JAXBException;
import javax.xml.bind.PropertyException;
import javax.xml.bind.UnmarshalException;
import javax.xml.bind.Unmarshaller;
import javax.xml.bind.ValidationEventHandler;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.attachment.AttachmentUnmarshaller;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLStreamReader;
import javax.xml.transform.Source;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import java.io.File;
import java.io.Reader;
import java.net.MalformedURLException;
import java.net.URL;

/**
 * Partial default <tt>Unmarshaller</tt> implementation.
 *
 * <p>
 * This class provides a partial default implementation for the
 * {@link javax.xml.bind.Unmarshaller}interface.
 *
 * <p>
 * A JAXB Provider has to implement five methods (getUnmarshallerHandler,
 * unmarshal(Node), unmarshal(XMLReader,InputSource),
 * unmarshal(XMLStreamReader), and unmarshal(XMLEventReader).
 *
 * @author <ul>
 *         <li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li>
 *         </ul>
 * @see javax.xml.bind.Unmarshaller
 * @since JAXB1.0
 */
public abstract class AbstractUnmarshallerImpl implements Unmarshaller
{
    /** handler that will be used to process errors and warnings during unmarshal */
    private ValidationEventHandler eventHandler =
        new DefaultValidationEventHandler();

    /** whether or not the unmarshaller will validate */
    protected boolean validating = false;
```

```java

    /**
     * XMLReader that will be used to parse a document.
     */
    private XMLReader reader = null;

    /**
     * Obtains a configured XMLReader.
     *
     * This method is used when the client-specified
     * {@link SAXSource} object doesn't have XMLReader.
     *
     * {@link Unmarshaller} is not re-entrant, so we will
     * only use one instance of XMLReader.
     */
    protected XMLReader getXMLReader() throws JAXBException {
        if(reader==null) {
            try {
                SAXParserFactory parserFactory;
                parserFactory = SAXParserFactory.newInstance();
                parserFactory.setNamespaceAware(true);
                // there is no point in asking a validation because
                // there is no guarantee that the document will come with
                // a proper schemaLocation.
                parserFactory.setValidating(false);
                reader = parserFactory.newSAXParser().getXMLReader();
            } catch( ParserConfigurationException e ) {
                throw new JAXBException(e);
            } catch( SAXException e ) {
                throw new JAXBException(e);
            }
        }
        return reader;
    }

    public Object unmarshal( Source source ) throws JAXBException {
        if( source == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "source" ) );
        }

        if(source instanceof SAXSource)
            return unmarshal( (SAXSource)source );
        if(source instanceof StreamSource)
            return unmarshal( streamSourceToInputSource((StreamSource)source));
        if(source instanceof DOMSource)
            return unmarshal( ((DOMSource)source).getNode() );

        // we don't handle other types of Source
        throw new IllegalArgumentException();
    }

    // use the client specified XMLReader contained in the SAXSource.
```

```
    private Object unmarshal( SAXSource source ) throws JAXBException {

        XMLReader r = source.getXMLReader();
        if( r == null )
            r = getXMLReader();

        return unmarshal( r, source.getInputSource() );
    }

    /**
     * Unmarshals an object by using the specified XMLReader and the InputSource.
     *
     * The callee should call the setErrorHandler method of the XMLReader
     * so that errors are passed to the client-specified ValidationEventHandler.
     */
    protected abstract Object unmarshal( XMLReader reader, InputSource source ) throws
        JAXBException;

    public final Object unmarshal( InputSource source ) throws JAXBException {
        if( source == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "source" ) );
        }

        return unmarshal( getXMLReader(), source );
    }


    private Object unmarshal( String url ) throws JAXBException {
        return unmarshal( new InputSource(url) );
    }

    public final Object unmarshal( URL url ) throws JAXBException {
        if( url == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "url" ) );
        }

        return unmarshal( url.toExternalForm() );
    }

    public final Object unmarshal( File f ) throws JAXBException {
        if( f == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "file" ) );
        }

        try {
            // copied from JAXP
            String path = f.getAbsolutePath();
            if (File.separatorChar != '/')
                path = path.replace(File.separatorChar, '/');
            if (!path.startsWith("/"))
```

```
                path = "/" + path;
            if (!path.endsWith("/") && f.isDirectory())
                path = path + "/";
            return unmarshal(new URL("file", "", path));
        } catch( MalformedURLException e ) {
            throw new IllegalArgumentException(e.getMessage());
        }
    }

    public final Object unmarshal( java.io.InputStream is )
        throws JAXBException {

        if( is == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "is" ) );
        }

        InputSource isrc = new InputSource( is );
        return unmarshal( isrc );
    }

    public final Object unmarshal( Reader reader ) throws JAXBException {
        if( reader == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "reader" ) );
        }

        InputSource isrc = new InputSource( reader );
        return unmarshal( isrc );
    }


    private static InputSource streamSourceToInputSource( StreamSource ss ) {
        InputSource is = new InputSource();
        is.setSystemId( ss.getSystemId() );
        is.setByteStream( ss.getInputStream() );
        is.setCharacterStream( ss.getReader() );

        return is;
    }


    /**
     * Indicates whether or not the Unmarshaller is configured to validate
     * during unmarshal operations.
     * <p>
     * <i><b>Note:</b> I named this method isValidating() to stay in-line
     * with JAXP, as opposed to naming it getValidating(). </i>
     *
     * @return true if the Unmarshaller is configured to validate during
     *        unmarshal operations, false otherwise
     * @throws JAXBException if an error occurs while retrieving the validating
     *        flag
```

```
     */
    public boolean isValidating() throws JAXBException {
        return validating;
    }

    /**
     * Allow an application to register a validation event handler.
     * <p>
     * The validation event handler will be called by the JAXB Provider if any
     * validation errors are encountered during calls to any of the
     * <tt>unmarshal</tt> methods.  If the client application does not register
     * a validation event handler before invoking the unmarshal methods, then
     * all validation events will be silently ignored and may result in
     * unexpected behaviour.
     *
     * @param handler the validation event handler
     * @throws JAXBException if an error was encountered while setting the
     *        event handler
     */
    public void setEventHandler(ValidationEventHandler handler)
        throws JAXBException {

        if( handler == null ) {
            eventHandler = new DefaultValidationEventHandler();
        } else {
            eventHandler = handler;
        }
    }

    /**
     * Specifies whether or not the Unmarshaller should validate during
     * unmarshal operations.  By default, the <tt>Unmarshaller</tt> does
     * not validate.
     * <p>
     * This method may only be invoked before or after calling one of the
     * unmarshal methods.
     *
     * @param validating true if the Unmarshaller should validate during
     *       unmarshal, false otherwise
     * @throws JAXBException if an error occurred while enabling or disabling
     * validation at unmarshal time
     */
    public void setValidating(boolean validating) throws JAXBException {
        this.validating = validating;
    }

    /**
     * Return the current event handler or the default event handler if one
     * hasn't been set.
     *
     * @return the current ValidationEventHandler or the default event handler
     *        if it hasn't been set
     * @throws JAXBException if an error was encountered while getting the
```

```
     *        current event handler
     */
    public ValidationEventHandler getEventHandler() throws JAXBException {
        return eventHandler;
    }


    /**
     * Creates an UnmarshalException from a SAXException.
     *
     * This is an utility method provided for the derived classes.
     *
     * <p>
     * When a provider-implemented ContentHandler wants to throw a
     * JAXBException, it needs to wrap the exception by a SAXException.
     * If the unmarshaller implementation blindly wrap SAXException
     * by JAXBException, such an exception will be a JAXBException
     * wrapped by a SAXException wrapped by another JAXBException.
     * This is silly.
     *
     * <p>
     * This method checks the nested exception of SAXException
     * and reduce those excessive wrapping.
     *
     * @return the resulting UnmarshalException
     */
    protected UnmarshalException createUnmarshalException( SAXException e ) {
        // check the nested exception to see if it's an UnmarshalException
        Exception nested = e.getException();
        if(nested instanceof UnmarshalException)
            return (UnmarshalException)nested;

        if(nested instanceof RuntimeException)
            // typically this is an unexpected exception,
            // just throw it rather than wrap it, so that the full stack
            // trace can be displayed.
            throw (RuntimeException)nested;


        // otherwise simply wrap it
        if(nested!=null)
            return new UnmarshalException(nested);
        else
            return new UnmarshalException(e);
    }

    /**
     * Default implementation of the setProperty method always
     * throws PropertyException since there are no required
     * properties. If a provider needs to handle additional
     * properties, it should override this method in a derived class.
     */
    public void setProperty( String name, Object value )
```

```
        throws PropertyException {

        if( name == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "name" ) );
        }

        throw new PropertyException(name, value);
    }

    /**
     * Default implementation of the getProperty method always
     * throws PropertyException since there are no required
     * properties. If a provider needs to handle additional
     * properties, it should override this method in a derived class.
     */
    public Object getProperty( String name )
        throws PropertyException {

        if( name == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "name" ) );
        }

        throw new PropertyException(name);
    }

    public Object unmarshal(XMLEventReader reader) throws JAXBException {

        throw new UnsupportedOperationException();
    }

    public Object unmarshal(XMLStreamReader reader) throws JAXBException {

        throw new UnsupportedOperationException();
    }

    public <T> JAXBElement<T> unmarshal(Node node, Class<T> expectedType) throws JAXBException {
        throw new UnsupportedOperationException();
    }

    public <T> JAXBElement<T> unmarshal(Source source, Class<T> expectedType) throws JAXBException
        {
        throw new UnsupportedOperationException();
    }

    public <T> JAXBElement<T> unmarshal(XMLStreamReader reader, Class<T> expectedType) throws
        JAXBException {
        throw new UnsupportedOperationException();
    }

    public <T> JAXBElement<T> unmarshal(XMLEventReader reader, Class<T> expectedType) throws
        JAXBException {
```

```
        throw new UnsupportedOperationException();
    }

    public void setSchema(Schema schema) {
        throw new UnsupportedOperationException();
    }

    public Schema getSchema() {
        throw new UnsupportedOperationException();
    }

    public void setAdapter(XmlAdapter adapter) {
        if(adapter==null)
            throw new IllegalArgumentException();
        setAdapter((Class)adapter.getClass(),adapter);
    }

    public <A extends XmlAdapter> void setAdapter(Class<A> type, A adapter) {
        throw new UnsupportedOperationException();
    }

    public <A extends XmlAdapter> A getAdapter(Class<A> type) {
        throw new UnsupportedOperationException();
    }

    public void setAttachmentUnmarshaller(AttachmentUnmarshaller au) {
        throw new UnsupportedOperationException();
    }

    public AttachmentUnmarshaller getAttachmentUnmarshaller() {
        throw new UnsupportedOperationException();
    }

    public void setListener(Listener listener) {
        throw new UnsupportedOperationException();
    }

    public Listener getListener() {
        throw new UnsupportedOperationException();
    }
}
```

6.3 DefaultValidationEventHandler.java

Secuencia 80: javax/xml/bind/helpers/DefaultValidationEventHandler.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import org.w3c.dom.Node;

import javax.xml.bind.ValidationEvent;
import javax.xml.bind.ValidationEventHandler;
import javax.xml.bind.ValidationEventLocator;
import java.net.URL;

/**
 * <p>
 * JAXB 1.0 only default validation event handler. This is the default
 * handler for all objects created from a JAXBContext that is managing
 * schema-derived code generated by a JAXB 1.0 binding compiler.
 *
 * <p>
 * This handler causes the unmarshal and validate operations to fail on the first
 * error or fatal error.
 *
 * <p>
 * This handler is not the default handler for JAXB mapped classes following
 * JAXB 2.0 or later versions. Default validation event handling has changed
 * and is specified in  {@link javax.xml.bind.Unmarshaller} and
 * {@link javax.xml.bind.Marshaller}.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.Unmarshaller
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @since JAXB1.0
 */
public class DefaultValidationEventHandler implements ValidationEventHandler {

    public boolean handleEvent( ValidationEvent event ) {

        if( event == null ) {
```

```
            throw new IllegalArgumentException();
        }

        // calculate the severity prefix and return value
        String severity = null;
        boolean retVal = false;
        switch ( event.getSeverity() ) {
            case ValidationEvent.WARNING:
                severity = Messages.format( Messages.WARNING );
                retVal = true; // continue after warnings
                break;
            case ValidationEvent.ERROR:
                severity = Messages.format( Messages.ERROR );
                retVal = false; // terminate after errors
                break;
            case ValidationEvent.FATAL_ERROR:
                severity = Messages.format( Messages.FATAL_ERROR );
                retVal = false; // terminate after fatal errors
                break;
            default:
                assert false :
                    Messages.format( Messages.UNRECOGNIZED_SEVERITY,
                            event.getSeverity() );
        }

        // calculate the location message
        String location = getLocation( event );

        System.out.println(
            Messages.format( Messages.SEVERITY_MESSAGE,
                             severity,
                             event.getMessage(),
                             location ) );

        // fail on the first error or fatal error
        return retVal;
    }

    /**
     * Calculate a location message for the event
     *
     */
    private String getLocation(ValidationEvent event) {
        StringBuffer msg = new StringBuffer();

        ValidationEventLocator locator = event.getLocator();

        if( locator != null ) {

            URL url = locator.getURL();
            Object obj = locator.getObject();
            Node node = locator.getNode();
            int line = locator.getLineNumber();
```

```

            if( url!=null || line!=-1 ) {
                msg.append( "line " + line );
                if( url!=null )
                    msg.append( " of " + url );
            } else if( obj != null ) {
                msg.append( " obj: " + obj.toString() );
            } else if( node != null ) {
                msg.append( " node: " + node.toString() );
            }
        } else {
            msg.append( Messages.format( Messages.LOCATION_UNAVAILABLE ) );
        }

        return msg.toString();
    }
}
```

6.4 Messages.java

Secuencia 81: javax/xml/bind/helpers/Messages.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import java.text.MessageFormat;
import java.util.ResourceBundle;

/**
 * Formats error messages.
```

```
 */
class Messages
{
    static String format( String property ) {
        return format( property, null );
    }

    static String format( String property, Object arg1 ) {
        return format( property, new Object[]{arg1} );
    }

    static String format( String property, Object arg1, Object arg2 ) {
        return format( property, new Object[]{arg1,arg2} );
    }

    static String format( String property, Object arg1, Object arg2, Object arg3 ) {
        return format( property, new Object[]{arg1,arg2,arg3} );
    }

    // add more if necessary.

    /** Loads a string resource and formats it with specified arguments. */
    static String format( String property, Object[] args ) {
        String text = ResourceBundle.getBundle(Messages.class.getName()).getString(property);
        return MessageFormat.format(text,args);
    }

//
//
// Message resources
//
//
    static final String INPUTSTREAM_NOT_NULL = // 0 args
        "AbstractUnmarshallerImpl.ISNotNull";

    static final String MUST_BE_BOOLEAN = // 1 arg
        "AbstractMarshallerImpl.MustBeBoolean";

    static final String MUST_BE_STRING = // 1 arg
        "AbstractMarshallerImpl.MustBeString";

    static final String SEVERITY_MESSAGE = // 3 args
        "DefaultValidationEventHandler.SeverityMessage";

    static final String LOCATION_UNAVAILABLE = // 0 args
        "DefaultValidationEventHandler.LocationUnavailable";

    static final String UNRECOGNIZED_SEVERITY = // 1 arg
        "DefaultValidationEventHandler.UnrecognizedSeverity";

    static final String WARNING = // 0 args
        "DefaultValidationEventHandler.Warning";

```

```
    static final String ERROR = // 0 args
        "DefaultValidationEventHandler.Error";

    static final String FATAL_ERROR = // 0 args
        "DefaultValidationEventHandler.FatalError";

    static final String ILLEGAL_SEVERITY = // 0 args
        "ValidationEventImpl.IllegalSeverity";

    static final String MUST_NOT_BE_NULL = // 1 arg
        "Shared.MustNotBeNull";
}
```

6.5 NotIdentifiableEventImpl.java

Secuencia 82: javax/xml/bind/helpers/NotIdentifiableEventImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import javax.xml.bind.ValidationEventLocator;

/**
 * Default implementation of the NotIdentifiableEvent interface.
 *
 * <p>
 * JAXB providers are allowed to use whatever class that implements
 * the ValidationEvent interface. This class is just provided for a
 * convenience.
 *
```

```
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.NotIdentifiableEvent
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class NotIdentifiableEventImpl
    extends ValidationEventImpl
    implements javax.xml.bind.NotIdentifiableEvent {

    /**
     * Create a new NotIdentifiableEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public NotIdentifiableEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator) {

        super(_severity, _message, _locator);
    }

    /**
     * Create a new NotIdentifiableEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @param _linkedException An optional linked exception that may provide
     * additional information about the event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public NotIdentifiableEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator,
                                     Throwable _linkedException) {

        super(_severity, _message, _locator, _linkedException);
    }

}
```

6.6 ParseConversionEventImpl.java

Secuencia 83: javax/xml/bind/helpers/ParseConversionEventImpl.java

```
/*
```

```
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import javax.xml.bind.ParseConversionEvent;
import javax.xml.bind.ValidationEventLocator;

/**
 * Default implementation of the ParseConversionEvent interface.
 *
 * <p>
 * JAXB providers are allowed to use whatever class that implements
 * the ValidationEvent interface. This class is just provided for a
 * convenience.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.ParseConversionEvent
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class ParseConversionEventImpl
    extends ValidationEventImpl
    implements ParseConversionEvent {

    /**
     * Create a new ParseConversionEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
```

```
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public ParseConversionEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator) {

        super(_severity, _message, _locator);
    }

    /**
     * Create a new ParseConversionEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @param _linkedException An optional linked exception that may provide
     * additional information about the event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public ParseConversionEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator,
                                     Throwable _linkedException) {

        super(_severity, _message, _locator, _linkedException);
    }

}
```

6.7 PrintConversionEventImpl.java

Secuencia 84: javax/xml/bind/helpers/PrintConversionEventImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import javax.xml.bind.PrintConversionEvent;
import javax.xml.bind.ValidationEventLocator;

/**
 * Default implementation of the PrintConversionEvent interface.
 *
 * <p>
 * JAXB providers are allowed to use whatever class that implements
 * the ValidationEvent interface. This class is just provided for a
 * convenience.
 *
 * @author <ul><li>Ryan Shoemaker, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.PrintConversionEvent
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class PrintConversionEventImpl
    extends ValidationEventImpl
    implements PrintConversionEvent {

    /**
     * Create a new PrintConversionEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public PrintConversionEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator) {

        super(_severity, _message, _locator);
    }

    /**
     * Create a new PrintConversionEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
```

```
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @param _linkedException An optional linked exception that may provide
     * additional information about the event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public PrintConversionEventImpl( int _severity, String _message,
                                     ValidationEventLocator _locator,
                                     Throwable _linkedException) {

        super(_severity, _message, _locator, _linkedException);
    }

}
```

6.8 ValidationEventImpl.java

Secuencia 85: javax/xml/bind/helpers/ValidationEventImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import java.text.MessageFormat;

import javax.xml.bind.ValidationEvent;
import javax.xml.bind.ValidationEventLocator;

/**
```

```
 * Default implementation of the ValidationEvent interface.
 *
 * <p>
 * JAXB providers are allowed to use whatever class that implements
 * the ValidationEvent interface. This class is just provided for a
 * convenience.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class ValidationEventImpl implements ValidationEvent
{

    /**
     * Create a new ValidationEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public ValidationEventImpl( int _severity, String _message,
                                ValidationEventLocator _locator ) {

        this(_severity,_message,_locator,null);
    }

    /**
     * Create a new ValidationEventImpl.
     *
     * @param _severity The severity value for this event.  Must be one of
     * ValidationEvent.WARNING, ValidationEvent.ERROR, or
     * ValidationEvent.FATAL_ERROR
     * @param _message The text message for this event - may be null.
     * @param _locator The locator object for this event - may be null.
     * @param _linkedException An optional linked exception that may provide
     * additional information about the event - may be null.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public ValidationEventImpl( int _severity, String _message,
                                ValidationEventLocator _locator,
                                Throwable _linkedException ) {

        setSeverity( _severity );
        this.message = _message;
        this.locator = _locator;
        this.linkedException = _linkedException;
```

```java
    }

    private int severity;
    private String message;
    private Throwable linkedException;
    private ValidationEventLocator locator;

    public int getSeverity() {
        return severity;
    }


    /**
     * Set the severity field of this event.
     *
     * @param _severity Must be one of ValidationEvent.WARNING,
     * ValidationEvent.ERROR, or ValidationEvent.FATAL_ERROR.
     * @throws IllegalArgumentException if an illegal severity field is supplied
     */
    public void setSeverity( int _severity ) {

        if( _severity != ValidationEvent.WARNING &&
            _severity != ValidationEvent.ERROR &&
            _severity != ValidationEvent.FATAL_ERROR ) {
                throw new IllegalArgumentException(
                    Messages.format( Messages.ILLEGAL_SEVERITY ) );
        }

        this.severity = _severity;
    }

    public String getMessage() {
        return message;
    }
    /**
     * Set the message field of this event.
     *
     * @param _message String message - may be null.
     */
    public void setMessage( String _message ) {
        this.message = _message;
    }

    public Throwable getLinkedException() {
        return linkedException;
    }
    /**
     * Set the linked exception field of this event.
     *
     * @param _linkedException Optional linked exception - may be null.
     */
    public void setLinkedException( Throwable _linkedException ) {
        this.linkedException = _linkedException;
```

```
    }

    public ValidationEventLocator getLocator() {
        return locator;
    }
    /**
     * Set the locator object for this event.
     *
     * @param _locator The locator - may be null.
     */
    public void setLocator( ValidationEventLocator _locator ) {
        this.locator = _locator;
    }

    /**
     * Returns a string representation of this object in a format
     * helpful to debugging.
     *
     * @see Object#equals(Object)
     */
    public String toString() {
        String s;
        switch(getSeverity()) {
        case WARNING:   s="WARNING";break;
        case ERROR: s="ERROR";break;
        case FATAL_ERROR: s="FATAL_ERROR";break;
        default: s=String.valueOf(getSeverity());break;
        }
        return MessageFormat.format("[severity={0},message={1},locator={2}]",
            new Object[]{
                s,
                getMessage(),
                getLocator()
            });
    }
}
```

6.9 ValidationEventLocatorImpl.java

Secuencia 86: javax/xml/bind/helpers/ValidationEventLocatorImpl.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.helpers;

import java.net.URL;
import java.net.MalformedURLException;
import java.text.MessageFormat;

import javax.xml.bind.ValidationEventLocator;
import org.w3c.dom.Node;
import org.xml.sax.Locator;
import org.xml.sax.SAXParseException;

/**
 * Default implementation of the ValidationEventLocator interface.
 *
 * <p>
 * JAXB providers are allowed to use whatever class that implements
 * the ValidationEventLocator interface. This class is just provided for a
 * convenience.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class ValidationEventLocatorImpl implements ValidationEventLocator
{
    /**
     * Creates an object with all fields unavailable.
     */
    public ValidationEventLocatorImpl() {
    }

    /**
     * Constructs an object from an org.xml.sax.Locator.
     *
     * The object's ColumnNumber, LineNumber, and URL become available from the
     * values returned by the locator's getColumnNumber(), getLineNumber(), and
     * getSystemId() methods respectively. Node, Object, and Offset are not
     * available.
```

```
     *
     * @param loc the SAX Locator object that will be used to populate this
     * event locator.
     * @throws IllegalArgumentException if the Locator is null
     */
    public ValidationEventLocatorImpl( Locator loc ) {
        if( loc == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "loc" ) );
        }

        this.url = toURL(loc.getSystemId());
        this.columnNumber = loc.getColumnNumber();
        this.lineNumber = loc.getLineNumber();
    }

    /**
     * Constructs an object from the location information of a SAXParseException.
     *
     * The object's ColumnNumber, LineNumber, and URL become available from the
     * values returned by the locator's getColumnNumber(), getLineNumber(), and
     * getSystemId() methods respectively. Node, Object, and Offset are not
     * available.
     *
     * @param e the SAXParseException object that will be used to populate this
     * event locator.
     * @throws IllegalArgumentException if the SAXParseException is null
     */
    public ValidationEventLocatorImpl( SAXParseException e ) {
        if( e == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "e" ) );
        }

        this.url = toURL(e.getSystemId());
        this.columnNumber = e.getColumnNumber();
        this.lineNumber = e.getLineNumber();
    }

    /**
     * Constructs an object that points to a DOM Node.
     *
     * The object's Node becomes available.  ColumnNumber, LineNumber, Object,
     * Offset, and URL are not available.
     *
     * @param _node the DOM Node object that will be used to populate this
     * event locator.
     * @throws IllegalArgumentException if the Node is null
     */
    public ValidationEventLocatorImpl(Node _node) {
        if( _node == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "_node" ) );
```

```
        }

        this.node = _node;
    }

    /**
     * Constructs an object that points to a JAXB content object.
     *
     * The object's Object becomes available. ColumnNumber, LineNumber, Node,
     * Offset, and URL are not available.
     *
     * @param _object the Object that will be used to populate this
     * event locator.
     * @throws IllegalArgumentException if the Object is null
     */
    public ValidationEventLocatorImpl(Object _object) {
        if( _object == null ) {
            throw new IllegalArgumentException(
                Messages.format( Messages.MUST_NOT_BE_NULL, "_object" ) );
        }

        this.object = _object;
    }

    /** Converts a system ID to an URL object. */
    private static URL toURL( String systemId ) {
        try {
            return new URL(systemId);
        } catch( MalformedURLException e ) {
            // TODO: how should we handle system id here?
            return null;    // for now
        }
    }

    private URL url = null;
    private int offset = -1;
    private int lineNumber = -1;
    private int columnNumber = -1;
    private Object object = null;
    private Node node = null;


    /**
     * @see javax.xml.bind.ValidationEventLocator#getURL()
     */
    public URL getURL() {
        return url;
    }

    /**
     * Set the URL field on this event locator.  Null values are allowed.
     *
     * @param _url the url
```

```
     */
    public void setURL( URL _url ) {
        this.url = _url;
    }

    /**
     * @see javax.xml.bind.ValidationEventLocator#getOffset()
     */
    public int getOffset() {
        return offset;
    }

    /**
     * Set the offset field on this event locator.
     *
     * @param _offset the offset
     */
    public void setOffset( int _offset ) {
        this.offset = _offset;
    }

    /**
     * @see javax.xml.bind.ValidationEventLocator#getLineNumber()
     */
    public int getLineNumber() {
        return lineNumber;
    }

    /**
     * Set the lineNumber field on this event locator.
     *
     * @param _lineNumber the line number
     */
    public void setLineNumber( int _lineNumber ) {
        this.lineNumber = _lineNumber;
    }

    /**
     * @see javax.xml.bind.ValidationEventLocator#getColumnNumber()
     */
    public int getColumnNumber() {
        return columnNumber;
    }

    /**
     * Set the columnNumber field on this event locator.
     *
     * @param _columnNumber the column number
     */
    public void setColumnNumber( int _columnNumber ) {
        this.columnNumber = _columnNumber;
    }

```

```java
    /**
     * @see javax.xml.bind.ValidationEventLocator#getObject()
     */
    public Object getObject() {
        return object;
    }

    /**
     * Set the Object field on this event locator.  Null values are allowed.
     *
     * @param _object the java content object
     */
    public void setObject( Object _object ) {
        this.object = _object;
    }

    /**
     * @see javax.xml.bind.ValidationEventLocator#getNode()
     */
    public Node getNode() {
        return node;
    }

    /**
     * Set the Node field on this event locator.  Null values are allowed.
     *
     * @param _node the Node
     */
    public void setNode( Node _node ) {
        this.node = _node;
    }

    /**
     * Returns a string representation of this object in a format
     * helpful to debugging.
     *
     * @see Object#equals(Object)
     */
    public String toString() {
        return MessageFormat.format("[node={0},object={1},url={2},line={3},col={4},offset={5}]",
            getNode(),
            getObject(),
            getURL(),
            String.valueOf(getLineNumber()),
            String.valueOf(getColumnNumber()),
            String.valueOf(getOffset()));
    }
}
```

7 Xml Bind Util (javax.xml.bind.util package)

7.1 JAXBResult.java

Secuencia 87: javax/xml/bind/util/JAXBResult.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.util;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import javax.xml.bind.UnmarshallerHandler;
import javax.xml.transform.sax.SAXResult;

/**
 * JAXP {@link javax.xml.transform.Result} implementation
 * that unmarshals a JAXB object.
 *
 * <p>
 * This utility class is useful to combine JAXB with
 * other Java/XML technologies.
 *
 * <p>
 * The following example shows how to use JAXB to unmarshal a document
 * resulting from an XSLT transformation.
 *
 * <blockquote>
 *    <pre>
 *       JAXBResult result = new JAXBResult(
 *         JAXBContext.newInstance("org.acme.foo") );
 *
 *       // set up XSLT transformation
 *       TransformerFactory tf = TransformerFactory.newInstance();
```

```
 *       Transformer t = tf.newTransformer(new StreamSource("test.xsl"));
 *
 *       // run transformation
 *       t.transform(new StreamSource("document.xml"),result);
 *
 *       // obtain the unmarshalled content tree
 *       Object o = result.getResult();
 *    </pre>
 * </blockquote>
 *
 * <p>
 * The fact that JAXBResult derives from SAXResult is an implementation
 * detail. Thus in general applications are strongly discouraged from
 * accessing methods defined on SAXResult.
 *
 * <p>
 * In particular it shall never attempt to call the setHandler,
 * setLexicalHandler, and setSystemId methods.
 *
 * @author
 *      Kohsuke Kawaguchi (kohsuke.kawaguchi@sun.com)
 */
public class JAXBResult extends SAXResult {

    /**
     * Creates a new instance that uses the specified
     * JAXBContext to unmarshal.
     *
     * @param context The JAXBContext that will be used to create the
     * necessary Unmarshaller.  This parameter must not be null.
     * @exception JAXBException if an error is encountered while creating the
     * JAXBResult or if the context parameter is null.
     */
    public JAXBResult( JAXBContext context ) throws JAXBException {
        this( ( context == null ) ? assertionFailed() : context.createUnmarshaller() );
    }

    /**
     * Creates a new instance that uses the specified
     * Unmarshaller to unmarshal an object.
     *
     * <p>
     * This JAXBResult object will use the specified Unmarshaller
     * instance. It is the caller's responsibility not to use the
     * same Unmarshaller for other purposes while it is being
     * used by this object.
     *
     * <p>
     * The primary purpose of this method is to allow the client
     * to configure Unmarshaller. Unless you know what you are doing,
     * it's easier and safer to pass a JAXBContext.
     *
     * @param _unmarshaller the unmarshaller.  This parameter must not be null.
```

```
     * @throws JAXBException if an error is encountered while creating the
     * JAXBResult or the Unmarshaller parameter is null.
     */
    public JAXBResult( Unmarshaller _unmarshaller ) throws JAXBException {
        if( _unmarshaller == null )
            throw new JAXBException(
                Messages.format( Messages.RESULT_NULL_UNMARSHALLER ) );

        this.unmarshallerHandler = _unmarshaller.getUnmarshallerHandler();

        super.setHandler(unmarshallerHandler);
    }

    /**
     * Unmarshaller that will be used to unmarshal
     * the input documents.
     */
    private final UnmarshallerHandler unmarshallerHandler;

    /**
     * Gets the unmarshalled object created by the transformation.
     *
     * @return
     *      Always return a non-null object.
     *
     * @exception IllegalStateException
     *  if this method is called before an object is unmarshalled.
     *
     * @exception JAXBException
     *      if there is any unmarshalling error.
     *      Note that the implementation is allowed to throw SAXException
     *      during the parsing when it finds an error.
     */
    public Object getResult() throws JAXBException {
        return unmarshallerHandler.getResult();
    }

    /**
     * Hook to throw exception from the middle of a contructor chained call
     * to this
     */
    private static Unmarshaller assertionFailed() throws JAXBException {
        throw new JAXBException( Messages.format( Messages.RESULT_NULL_CONTEXT ) );
    }
}
```

7.2 JAXBSource.java

Secuencia 88: javax/xml/bind/util/JAXBSource.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.util;

import org.xml.sax.ContentHandler;
import org.xml.sax.DTDHandler;
import org.xml.sax.EntityResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.ext.LexicalHandler;
import org.xml.sax.helpers.XMLFilterImpl;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.transform.sax.SAXSource;
import org.xml.sax.XMLFilter;

/**
 * JAXP {@link javax.xml.transform.Source} implementation
 * that marshals a JAXB-generated object.
 *
 * <p>
 * This utility class is useful to combine JAXB with
 * other Java/XML technologies.
 *
 * <p>
 * The following example shows how to use JAXB to marshal a document
 * for transformation by XSLT.
 *
```

```
 * <blockquote>
 *    <pre>
 *       MyObject o = // get JAXB content tree
 *
 *       // jaxbContext is a JAXBContext object from which 'o' is created.
 *       JAXBSource source = new JAXBSource( jaxbContext, o );
 *
 *       // set up XSLT transformation
 *       TransformerFactory tf = TransformerFactory.newInstance();
 *       Transformer t = tf.newTransformer(new StreamSource("test.xsl"));
 *
 *       // run transformation
 *       t.transform(source,new StreamResult(System.out));
 *    </pre>
 * </blockquote>
 *
 * <p>
 * The fact that JAXBSource derives from SAXSource is an implementation
 * detail. Thus in general applications are strongly discouraged from
 * accessing methods defined on SAXSource. In particular,
 * the setXMLReader and setInputSource methods shall never be called.
 * The XMLReader object obtained by the getXMLReader method shall
 * be used only for parsing the InputSource object returned by
 * the getInputSource method.
 *
 * <p>
 * Similarly the InputSource object obtained by the getInputSource
 * method shall be used only for being parsed by the XMLReader object
 * returned by the getXMLReader.
 *
 * @author
 *      Kohsuke Kawaguchi (kohsuke.kawaguchi@sun.com)
 */
public class JAXBSource extends SAXSource {

    /**
     * Creates a new {@link javax.xml.transform.Source} for the given content object.
     *
     * @param   context
     *      JAXBContext that was used to create
     *      <code>contentObject</code>. This context is used
     *      to create a new instance of marshaller and must not be null.
     * @param   contentObject
     *      An instance of a JAXB-generated class, which will be
     *      used as a {@link javax.xml.transform.Source} (by marshalling it into XML).  It must
     *      not be null.
     * @throws JAXBException if an error is encountered while creating the
     * JAXBSource or if either of the parameters are null.
     */
    public JAXBSource( JAXBContext context, Object contentObject )
        throws JAXBException {

        this(
```

```
            ( context == null ) ?
                assertionFailed( Messages.format( Messages.SOURCE_NULL_CONTEXT ) ) :
                context.createMarshaller(),

            ( contentObject == null ) ?
                assertionFailed( Messages.format( Messages.SOURCE_NULL_CONTENT ) ) :
                contentObject);
    }

    /**
     * Creates a new {@link javax.xml.transform.Source} for the given content object.
     *
     * @param   marshaller
     *      A marshaller instance that will be used to marshal
     *      <code>contentObject</code> into XML. This must be
     *      created from a JAXBContext that was used to build
     *      <code>contentObject</code> and must not be null.
     * @param   contentObject
     *      An instance of a JAXB-generated class, which will be
     *      used as a {@link javax.xml.transform.Source} (by marshalling it into XML).  It must
     *      not be null.
     * @throws JAXBException if an error is encountered while creating the
     * JAXBSource or if either of the parameters are null.
     */
    public JAXBSource( Marshaller marshaller, Object contentObject )
        throws JAXBException {

        if( marshaller == null )
            throw new JAXBException(
                Messages.format( Messages.SOURCE_NULL_MARSHALLER ) );

        if( contentObject == null )
            throw new JAXBException(
                Messages.format( Messages.SOURCE_NULL_CONTENT ) );

        this.marshaller = marshaller;
        this.contentObject = contentObject;

        super.setXMLReader(pseudoParser);
        // pass a dummy InputSource. We don't care
        super.setInputSource(new InputSource());
    }

    private final Marshaller marshaller;
    private final Object contentObject;

    // this object will pretend as an XMLReader.
    // no matter what parameter is specified to the parse method,
    // it just parse the contentObject.
    private final XMLReader pseudoParser = new XMLReader() {
        public boolean getFeature(String name) throws SAXNotRecognizedException {
            if(name.equals("http://xml.org/sax/features/namespaces"))
                return true;
```

```
            if(name.equals("http://xml.org/sax/features/namespace-prefixes"))
                return false;
            throw new SAXNotRecognizedException(name);
        }

        public void setFeature(String name, boolean value) throws SAXNotRecognizedException {
            if(name.equals("http://xml.org/sax/features/namespaces") && value)
                return;
            if(name.equals("http://xml.org/sax/features/namespace-prefixes") && !value)
                return;
            throw new SAXNotRecognizedException(name);
        }

        public Object getProperty(String name) throws SAXNotRecognizedException {
            if( "http://xml.org/sax/properties/lexical-handler".equals(name) ) {
                return lexicalHandler;
            }
            throw new SAXNotRecognizedException(name);
        }

        public void setProperty(String name, Object value) throws SAXNotRecognizedException {
            if( "http://xml.org/sax/properties/lexical-handler".equals(name) ) {
                this.lexicalHandler = (LexicalHandler)value;
                return;
            }
            throw new SAXNotRecognizedException(name);
        }

        private LexicalHandler lexicalHandler;

        // we will store this value but never use it by ourselves.
        private EntityResolver entityResolver;
        public void setEntityResolver(EntityResolver resolver) {
            this.entityResolver = resolver;
        }
        public EntityResolver getEntityResolver() {
            return entityResolver;
        }

        private DTDHandler dtdHandler;
        public void setDTDHandler(DTDHandler handler) {
            this.dtdHandler = handler;
        }
        public DTDHandler getDTDHandler() {
            return dtdHandler;
        }

        // SAX allows ContentHandler to be changed during the parsing,
        // but JAXB doesn't. So this repeater will sit between those
        // two components.
        private XMLFilter repeater = new XMLFilterImpl();

        public void setContentHandler(ContentHandler handler) {
```

```
            repeater.setContentHandler(handler);
        }
        public ContentHandler getContentHandler() {
            return repeater.getContentHandler();
        }

        private ErrorHandler errorHandler;
        public void setErrorHandler(ErrorHandler handler) {
            this.errorHandler = handler;
        }
        public ErrorHandler getErrorHandler() {
            return errorHandler;
        }

        public void parse(InputSource input) throws SAXException {
            parse();
        }

        public void parse(String systemId) throws SAXException {
            parse();
        }

        public void parse() throws SAXException {
            // parses a content object by using the given marshaller
            // SAX events will be sent to the repeater, and the repeater
            // will further forward it to an appropriate component.
            try {
                marshaller.marshal( contentObject, (XMLFilterImpl)repeater );
            } catch( JAXBException e ) {
                // wrap it to a SAXException
                SAXParseException se =
                    new SAXParseException( e.getMessage(),
                        null, null, -1, -1, e );

                // if the consumer sets an error handler, it is our responsibility
                // to notify it.
                if(errorHandler!=null)
                    errorHandler.fatalError(se);

                // this is a fatal error. Even if the error handler
                // returns, we will abort anyway.
                throw se;
            }
        }
    };

    /**
     * Hook to throw exception from the middle of a contructor chained call
     * to this
     */
    private static Marshaller assertionFailed( String message )
        throws JAXBException {

```

```
        throw new JAXBException( message );
    }
}
```

7.3 Messages.java

Secuencia 89: javax/xml/bind/util/Messages.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.util;

import java.text.MessageFormat;
import java.util.ResourceBundle;

/**
 * Formats error messages.
 */
class Messages
{
    static String format( String property ) {
        return format( property, null );
    }

    static String format( String property, Object arg1 ) {
        return format( property, new Object[]{arg1} );
    }

    static String format( String property, Object arg1, Object arg2 ) {
        return format( property, new Object[]{arg1,arg2} );
    }
```

```

    static String format( String property, Object arg1, Object arg2, Object arg3 ) {
        return format( property, new Object[]{arg1,arg2,arg3} );
    }

    // add more if necessary.

    /** Loads a string resource and formats it with specified arguments. */
    static String format( String property, Object[] args ) {
        String text = ResourceBundle.getBundle(Messages.class.getName()).getString(property);
        return MessageFormat.format(text,args);
    }

//
//
// Message resources
//
//
    static final String UNRECOGNIZED_SEVERITY = // 1 arg
        "ValidationEventCollector.UnrecognizedSeverity";

    static final String RESULT_NULL_CONTEXT = // 0 args
        "JAXBResult.NullContext";

    static final String RESULT_NULL_UNMARSHALLER = // 0 arg
        "JAXBResult.NullUnmarshaller";

    static final String SOURCE_NULL_CONTEXT = // 0 args
        "JAXBSource.NullContext";

    static final String SOURCE_NULL_CONTENT = // 0 arg
        "JAXBSource.NullContent";

    static final String SOURCE_NULL_MARSHALLER = // 0 arg
        "JAXBSource.NullMarshaller";

}
```

7.4 ValidationEventCollector.java

Secuencia 90: javax/xml/bind/util/ValidationEventCollector.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.bind.util;

import javax.xml.bind.ValidationEvent;
import javax.xml.bind.ValidationEventHandler;
import java.util.ArrayList;
import java.util.List;

/**
 * {@link javax.xml.bind.ValidationEventHandler ValidationEventHandler}
 * implementation that collects all events.
 *
 * <p>
 * To use this class, create a new instance and pass it to the setEventHandler
 * method of the Validator, Unmarshaller, Marshaller class.  After the call to
 * validate or unmarshal completes, call the getEvents method to retrieve all
 * the reported errors and warnings.
 *
 * @author <ul><li>Kohsuke Kawaguchi, Sun Microsystems, Inc.</li><li>Ryan Shoemaker, Sun
     Microsystems, Inc.</li><li>Joe Fialli, Sun Microsystems, Inc.</li></ul>
 * @see javax.xml.bind.Validator
 * @see javax.xml.bind.ValidationEventHandler
 * @see javax.xml.bind.ValidationEvent
 * @see javax.xml.bind.ValidationEventLocator
 * @since JAXB1.0
 */
public class ValidationEventCollector implements ValidationEventHandler
{
    private final List<ValidationEvent> events = new ArrayList<ValidationEvent>();

    /**
     * Return an array of ValidationEvent objects containing a copy of each of
     * the collected errors and warnings.
     *
     * @return
     *      a copy of all the collected errors and warnings or an empty array
     *      if there weren't any
     */
    public ValidationEvent[] getEvents() {
        return events.toArray(new ValidationEvent[events.size()]);
    }
```

```java

    /**
     * Clear all collected errors and warnings.
     */
    public void reset() {
        events.clear();
    }

    /**
     * Returns true if this event collector contains at least one
     * ValidationEvent.
     *
     * @return true if this event collector contains at least one
     *         ValidationEvent, false otherwise
     */
    public boolean hasEvents() {
        return !events.isEmpty();
    }

    public boolean handleEvent( ValidationEvent event ) {
        events.add(event);

        boolean retVal = true;
        switch( event.getSeverity() ) {
            case ValidationEvent.WARNING:
                retVal = true; // continue validation
                break;
            case ValidationEvent.ERROR:
                retVal = true; // continue validation
                break;
            case ValidationEvent.FATAL_ERROR:
                retVal = false; // halt validation
                break;
            default:
                _assert( false,
                         Messages.format( Messages.UNRECOGNIZED_SEVERITY,
                                 event.getSeverity() ) );
                break;
        }

        return retVal;
    }

    private static void _assert( boolean b, String msg ) {
        if( !b ) {
            throw new InternalError( msg );
        }
    }
}
```

8 Xml Crypto (javax.xml.crypto package)

8.1 AlgorithmMethod.java

Secuencia 91: javax/xml/crypto/AlgorithmMethod.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: AlgorithmMethod.java,v 1.4 2005/05/10 15:47:41 mullan Exp $
 */
package javax.xml.crypto;

import java.security.spec.AlgorithmParameterSpec;

/**
 * An abstract representation of an algorithm defined in the XML Security
 * specifications. Subclasses represent specific types of XML security
 * algorithms, such as a {@link javax.xml.crypto.dsig.Transform}.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface AlgorithmMethod {

    /**
     * Returns the algorithm URI of this <code>AlgorithmMethod</code>.
     *
     * @return the algorithm URI of this <code>AlgorithmMethod</code>
     */
```

```
    String getAlgorithm();

    /**
     * Returns the algorithm parameters of this <code>AlgorithmMethod</code>.
     *
     * @return the algorithm parameters of this <code>AlgorithmMethod</code>.
     *    Returns <code>null</code> if this <code>AlgorithmMethod</code> does
     *    not require parameters and they are not specified.
     */
    AlgorithmParameterSpec getParameterSpec();
}
```

8.2 Data.java

Secuencia 92: javax/xml/crypto/Data.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: Data.java,v 1.4 2005/05/10 15:47:41 mullan Exp $
 */
package javax.xml.crypto;

import javax.xml.crypto.dsig.Transform;

/**
 * An abstract representation of the result of dereferencing a
 * {@link URIReference} or the input/output of subsequent {@link Transform}s.
 * The primary purpose of this interface is to group and provide type safety
 * for all <code>Data</code> subtypes.
 *
 * @author Sean Mullan
```

```
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface Data { }
```

8.3 KeySelector.java

Secuencia 93: javax/xml/crypto/KeySelector.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeySelector.java,v 1.6 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.security.Key;
import javax.xml.crypto.dsig.keyinfo.KeyInfo;
import javax.xml.crypto.dsig.keyinfo.RetrievalMethod;

/**
 * A selector that finds and returns a key using the data contained in a
 * {@link KeyInfo} object. An example of an implementation of
 * this class is one that searches a {@link java.security.KeyStore} for
 * trusted keys that match information contained in a <code>KeyInfo</code>.
 *
 * <p>Whether or not the returned key is trusted and the mechanisms
 * used to determine that is implementation-specific.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
```

```
 */
public abstract class KeySelector {

    /**
     * The purpose of the key that is to be selected.
     */
    public static class Purpose {

        private final String name;

        private Purpose(String name)    { this.name = name; }

        /**
         * Returns a string representation of this purpose ("sign",
         * "verify", "encrypt", or "decrypt").
         *
         * @return a string representation of this purpose
         */
        public String toString()        { return name; }

        /**
         * A key for signing.
         */
        public static final Purpose SIGN = new Purpose("sign");
        /**
         * A key for verifying.
         */
        public static final Purpose VERIFY = new Purpose("verify");
        /**
         * A key for encrypting.
         */
        public static final Purpose ENCRYPT = new Purpose("encrypt");
        /**
         * A key for decrypting.
         */
        public static final Purpose DECRYPT = new Purpose("decrypt");
    }

    /**
     * Default no-args constructor; intended for invocation by subclasses only.
     */
    protected KeySelector() {}

    /**
     * Attempts to find a key that satisfies the specified constraints.
     *
     * @param keyInfo a <code>KeyInfo</code> (may be <code>null</code>)
     * @param purpose the key's purpose ({@link Purpose#SIGN},
     *    {@link Purpose#VERIFY}, {@link Purpose#ENCRYPT}, or
     *    {@link Purpose#DECRYPT})
     * @param method the algorithm method that this key is to be used for.
     *    Only keys that are compatible with the algorithm and meet the
     *    constraints of the specified algorithm should be returned.
```

```
     * @param context an <code>XMLCryptoContext</code> that may contain
     *    useful information for finding an appropriate key. If this key
     *    selector supports resolving {@link RetrievalMethod} types, the
     *    context's <code>baseURI</code> and <code>dereferencer</code>
     *    parameters (if specified) should be used by the selector to
     *    resolve and dereference the URI.
     * @return the result of the key selector
     * @throws KeySelectorException if an exceptional condition occurs while
     *    attempting to find a key. Note that an inability to find a key is not
     *    considered an exception (<code>null</code> should be
     *    returned in that case). However, an error condition (ex: network
     *    communications failure) that prevented the <code>KeySelector</code>
     *    from finding a potential key should be considered an exception.
     * @throws ClassCastException if the data type of <code>method</code>
     *    is not supported by this key selector
     */
    public abstract KeySelectorResult select(KeyInfo keyInfo, Purpose purpose,
        AlgorithmMethod method, XMLCryptoContext context)
        throws KeySelectorException;

    /**
     * Returns a <code>KeySelector</code> that always selects the specified
     * key, regardless of the <code>KeyInfo</code> passed to it.
     *
     * @param key the sole key to be stored in the key selector
     * @return a key selector that always selects the specified key
     * @throws NullPointerException if <code>key</code> is <code>null</code>
     */
    public static KeySelector singletonKeySelector(Key key) {
        return new SingletonKeySelector(key);
    }

    private static class SingletonKeySelector extends KeySelector {
        private final Key key;

        SingletonKeySelector(Key key) {
            if (key == null) {
                throw new NullPointerException();
            }
            this.key = key;
        }

        public KeySelectorResult select(KeyInfo keyInfo, Purpose purpose,
            AlgorithmMethod method, XMLCryptoContext context)
            throws KeySelectorException {

            return new KeySelectorResult() {
                public Key getKey() {
                    return key;
                }
            };
        }
    }
```

```
}
```

8.4 KeySelectorException.java

Secuencia 94: javax/xml/crypto/KeySelectorException.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeySelectorException.java,v 1.3 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.io.PrintStream;
import java.io.PrintWriter;

/**
 * Indicates an exceptional condition thrown by a {@link KeySelector}.
 *
 * <p>A <code>KeySelectorException</code> can contain a cause: another
 * throwable that caused this <code>KeySelectorException</code> to get thrown.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public class KeySelectorException extends Exception {

    private static final long serialVersionUID = -7480033639322531109L;

    /**
     * The throwable that caused this exception to get thrown, or
```

```
     * <code>null</code> if this exception was not caused by another throwable
     * or if the causative throwable is unknown.
     *
     * @serial
     */
    private Throwable cause;

    /**
     * Constructs a new <code>KeySelectorException</code> with
     * <code>null</code> as its detail message.
     */
    public KeySelectorException() {
        super();
    }

    /**
     * Constructs a new <code>KeySelectorException</code> with the specified
     * detail message.
     *
     * @param message the detail message
     */
    public KeySelectorException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>KeySelectorException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public KeySelectorException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>KeySelectorException</code> with the specified
     * cause and a detail message of
     * <code>(cause==null ? null : cause.toString())</code>
     * (which typically contains the class and detail message of
     * <code>cause</code>).
     *
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public KeySelectorException(Throwable cause) {
        super(cause==null ? null : cause.toString());
```

```
        this.cause = cause;
    }

    /**
     * Returns the cause of this <code>KeySelectorException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>KeySelectorException</code> to get thrown.)
     *
     * @return the cause of this <code>KeySelectorException</code> or
     *         <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>KeySelectorException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
     */
    public void printStackTrace() {
        super.printStackTrace();
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>KeySelectorException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>KeySelectorException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }
}
```

8.5 KeySelectorResult.java

Secuencia 95: javax/xml/crypto/KeySelectorResult.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeySelectorResult.java,v 1.3 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.security.Key;

/**
 * The result returned by the {@link KeySelector#select KeySelector.select}
 * method.
 * <p>
 * At a minimum, a <code>KeySelectorResult</code> contains the <code>Key</code>
 * selected by the <code>KeySelector</code>. Implementations of this interface
 * may add methods to return implementation or algorithm specific information,
 * such as a chain of certificates or debugging information.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeySelector
 */
public interface KeySelectorResult {

    /**
     * Returns the selected key.
     *
     * @return the selected key, or <code>null</code> if none can be found
     */
    Key getKey();
}
```

8.6 MarshalException.java

Secuencia 96: javax/xml/crypto/MarshalException.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: MarshalException.java,v 1.5 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.io.PrintStream;
import java.io.PrintWriter;
import javax.xml.crypto.dsig.Manifest;
import javax.xml.crypto.dsig.XMLSignature;
import javax.xml.crypto.dsig.XMLSignatureFactory;
import javax.xml.crypto.dsig.keyinfo.KeyInfo;
import javax.xml.crypto.dsig.keyinfo.KeyInfoFactory;

/**
 * Indicates an exceptional condition that occurred during the XML
 * marshalling or unmarshalling process.
 *
 * <p>A <code>MarshalException</code> can contain a cause: another
 * throwable that caused this <code>MarshalException</code> to get thrown.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignature#sign(XMLSignContext)
 * @see XMLSignatureFactory#unmarshalXMLSignature(XMLValidateContext)
 */
```

```java
public class MarshalException extends Exception {

    private static final long serialVersionUID = -863185580332643547L;

    /**
     * The throwable that caused this exception to get thrown, or null if this
     * exception was not caused by another throwable or if the causative
     * throwable is unknown.
     *
     * @serial
     */
    private Throwable cause;

    /**
     * Constructs a new <code>MarshalException</code> with
     * <code>null</code> as its detail message.
     */
    public MarshalException() {
        super();
    }

    /**
     * Constructs a new <code>MarshalException</code> with the specified
     * detail message.
     *
     * @param message the detail message
     */
    public MarshalException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>MarshalException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public MarshalException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>MarshalException</code> with the specified cause
     * and a detail message of <code>(cause==null ? null : cause.toString())
     * </code> (which typically contains the class and detail message of
     * <code>cause</code>).
     *
```

```
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public MarshalException(Throwable cause) {
        super(cause==null ? null : cause.toString());
        this.cause = cause;
    }

    /**
     * Returns the cause of this <code>MarshalException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>MarshalException</code> to get thrown.)
     *
     * @return the cause of this <code>MarshalException</code> or
     *         <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>MarshalException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
     */
    public void printStackTrace() {
        super.printStackTrace();
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>MarshalException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>MarshalException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }
}
```

8.7 NoSuchMechanismException.java

Secuencia 97: javax/xml/crypto/NoSuchMechanismException.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: NoSuchMechanismException.java,v 1.4 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.io.PrintStream;
import java.io.PrintWriter;
import javax.xml.crypto.dsig.Manifest;
import javax.xml.crypto.dsig.XMLSignature;
import javax.xml.crypto.dsig.XMLSignatureFactory;
import javax.xml.crypto.dsig.keyinfo.KeyInfo;
import javax.xml.crypto.dsig.keyinfo.KeyInfoFactory;

/**
 * This exception is thrown when a particular XML mechanism is requested but
 * is not available in the environment.
 *
 * <p>A <code>NoSuchMechanismException</code> can contain a cause: another
 * throwable that caused this <code>NoSuchMechanismException</code> to get
 * thrown.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#getInstance XMLSignatureFactory.getInstance
 * @see KeyInfoFactory#getInstance KeyInfoFactory.getInstance
```

```
 */
public class NoSuchMechanismException extends RuntimeException {

    private static final long serialVersionUID = 4189669069570660166L;

    /**
     * The throwable that caused this exception to get thrown, or null if this
     * exception was not caused by another throwable or if the causative
     * throwable is unknown.
     *
     * @serial
     */
    private Throwable cause;

    /**
     * Constructs a new <code>NoSuchMechanismException</code> with
     * <code>null</code> as its detail message.
     */
    public NoSuchMechanismException() {
        super();
    }

    /**
     * Constructs a new <code>NoSuchMechanismException</code> with the
     * specified detail message.
     *
     * @param message the detail message
     */
    public NoSuchMechanismException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>NoSuchMechanismException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public NoSuchMechanismException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>NoSuchMechanismException</code> with the
     * specified cause and a detail message of
     * <code>(cause==null ? null : cause.toString())</code> (which typically
     * contains the class and detail message of <code>cause</code>).
```

```
     *
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public NoSuchMechanismException(Throwable cause) {
        super(cause==null ? null : cause.toString());
        this.cause = cause;
    }

    /**
     * Returns the cause of this <code>NoSuchMechanismException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>NoSuchMechanismException</code> to get thrown.)
     *
     * @return the cause of this <code>NoSuchMechanismException</code> or
     *         <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>NoSuchMechanismException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
     */
    public void printStackTrace() {
        super.printStackTrace();
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>NoSuchMechanismException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>NoSuchMechanismException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }
}
```

8.8 NodeSetData.java

Secuencia 98: javax/xml/crypto/NodeSetData.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: NodeSetData.java,v 1.5 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.util.Iterator;

/**
 * An abstract representation of a <code>Data</code> type containing a
 * node-set. The type (class) and ordering of the nodes contained in the set
 * are not defined by this class; instead that behavior should be
 * defined by <code>NodeSetData</code> subclasses.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface NodeSetData extends Data {

    /**
     * Returns a read-only iterator over the nodes contained in this
     * <code>NodeSetData</code> in
     * <a href="http://www.w3.org/TR/1999/REC-xpath-19991116#dt-document-order">
     * document order</a>. Attempts to modify the returned iterator
     * via the <code>remove</code> method throw
     * <code>UnsupportedOperationException</code>.
```

```
     *
     * @return an <code>Iterator</code> over the nodes in this
     *    <code>NodeSetData</code> in document order
     */
    @SuppressWarnings("rawtypes")
    Iterator iterator();
}
```

8.9 OctetStreamData.java

Secuencia 99: javax/xml/crypto/OctetStreamData.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: OctetStreamData.java,v 1.3 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.io.InputStream;

/**
 * A representation of a <code>Data</code> type containing an octet stream.
 *
 * @since 1.6
 */
public class OctetStreamData implements Data {

    private InputStream octetStream;
    private String uri;
    private String mimeType;

```

```java
    /**
     * Creates a new <code>OctetStreamData</code>.
     *
     * @param octetStream the input stream containing the octets
     * @throws NullPointerException if <code>octetStream</code> is
     *    <code>null</code>
     */
    public OctetStreamData(InputStream octetStream) {
        if (octetStream == null) {
            throw new NullPointerException("octetStream is null");
        }
        this.octetStream = octetStream;
    }

    /**
     * Creates a new <code>OctetStreamData</code>.
     *
     * @param octetStream the input stream containing the octets
     * @param uri the URI String identifying the data object (may be
     *    <code>null</code>)
     * @param mimeType the MIME type associated with the data object (may be
     *    <code>null</code>)
     * @throws NullPointerException if <code>octetStream</code> is
     *    <code>null</code>
     */
    public OctetStreamData(InputStream octetStream, String uri,
        String mimeType) {
        if (octetStream == null) {
            throw new NullPointerException("octetStream is null");
        }
        this.octetStream = octetStream;
        this.uri = uri;
        this.mimeType = mimeType;
    }

    /**
     * Returns the input stream of this <code>OctetStreamData</code>.
     *
     * @return the input stream of this <code>OctetStreamData</code>.
     */
    public InputStream getOctetStream() {
        return octetStream;
    }

    /**
     * Returns the URI String identifying the data object represented by this
     * <code>OctetStreamData</code>.
     *
     * @return the URI String or <code>null</code> if not applicable
     */
    public String getURI() {
        return uri;
    }
```

```

    /**
     * Returns the MIME type associated with the data object represented by this
     * <code>OctetStreamData</code>.
     *
     * @return the MIME type or <code>null</code> if not applicable
     */
    public String getMimeType() {
        return mimeType;
    }
}
```

8.10 URIDereferencer.java

Secuencia 100: javax/xml/crypto/URIDereferencer.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * ============================================================================
 *
 * (C) Copyright IBM Corp. 2003 All Rights Reserved.
 *
 * ============================================================================
 */
/*
 * $Id: URIDereferencer.java,v 1.5 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

/**
```

```
 * A dereferencer of {@link URIReference}s.
 * <p>
 * The result of dereferencing a <code>URIReference</code> is either an
 * instance of {@link OctetStreamData} or {@link NodeSetData}. Unless the
 * <code>URIReference</code> is a <i>same-document reference</i> as defined
 * in section 4.2 of the W3C Recommendation for XML-Signature Syntax and
 * Processing, the result of dereferencing the <code>URIReference</code>
 * MUST be an <code>OctetStreamData</code>.
 *
 * @author Sean Mullan
 * @author Joyce Leung
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLCryptoContext#setURIDereferencer(URIDereferencer)
 * @see XMLCryptoContext#getURIDereferencer
 */
public interface URIDereferencer {

    /**
     * Dereferences the specified <code>URIReference</code> and returns the
     * dereferenced data.
     *
     * @param uriReference the <code>URIReference</code>
     * @param context an <code>XMLCryptoContext</code> that may
     *    contain additional useful information for dereferencing the URI. This
     *    implementation should dereference the specified
     *    <code>URIReference</code> against the context's <code>baseURI</code>
     *    parameter, if specified.
     * @return the dereferenced data
     * @throws NullPointerException if <code>uriReference</code> or
     *    <code>context</code> are <code>null</code>
     * @throws URIReferenceException if an exception occurs while
     *    dereferencing the specified <code>uriReference</code>
     */
    Data dereference(URIReference uriReference, XMLCryptoContext context)
        throws URIReferenceException;
}
```

8.11 URIReference.java

Secuencia 101: javax/xml/crypto/URIReference.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: URIReference.java,v 1.4 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

/**
 * Identifies a data object via a URI-Reference, as specified by
 * <a href="http://www.ietf.org/rfc/rfc2396.txt">RFC 2396</a>.
 *
 * <p>Note that some subclasses may not have a <code>type</code> attribute
 * and for objects of those types, the {@link #getType} method always returns
 * <code>null</code>.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see URIDereferencer
 */
public interface URIReference {

    /**
     * Returns the URI of the referenced data object.
     *
     * @return the URI of the data object in RFC 2396 format (may be
     *    <code>null</code> if not specified)
     */
    String getURI();

    /**
     * Returns the type of data referenced by this URI.
     *
     * @return the type (a URI) of the data object (may be <code>null</code>
     *    if not specified)
     */
    String getType();
}
```

8.12 URIReferenceException.java

Secuencia 102: javax/xml/crypto/URIReferenceException.java

```
/*
```

```
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: URIReferenceException.java,v 1.4 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

import java.io.PrintStream;
import java.io.PrintWriter;
import javax.xml.crypto.dsig.keyinfo.RetrievalMethod;

/**
 * Indicates an exceptional condition thrown while dereferencing a
 * {@link URIReference}.
 *
 * <p>A <code>URIReferenceException</code> can contain a cause: another
 * throwable that caused this <code>URIReferenceException</code> to get thrown.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see URIDereferencer#dereference(URIReference, XMLCryptoContext)
 * @see RetrievalMethod#dereference(XMLCryptoContext)
 */
public class URIReferenceException extends Exception {

    private static final long serialVersionUID = 7173469703932561419L;

    /**
     * The throwable that caused this exception to get thrown, or null if this
     * exception was not caused by another throwable or if the causative
     * throwable is unknown.
```

```
     *
     * @serial
     */
    private Throwable cause;

    private URIReference uriReference;

    /**
     * Constructs a new <code>URIReferenceException</code> with
     * <code>null</code> as its detail message.
     */
    public URIReferenceException() {
        super();
    }

    /**
     * Constructs a new <code>URIReferenceException</code> with the specified
     * detail message.
     *
     * @param message the detail message
     */
    public URIReferenceException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>URIReferenceException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public URIReferenceException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>URIReferenceException</code> with the
     * specified detail message, cause and <code>URIReference</code>.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     * @param uriReference the <code>URIReference</code> that was being
     *    dereferenced when the error was encountered
```

```
     * @throws NullPointerException if <code>uriReference</code> is
     *    <code>null</code>
     */
    public URIReferenceException(String message, Throwable cause,
        URIReference uriReference) {
        this(message, cause);
        if (uriReference == null) {
            throw new NullPointerException("uriReference cannot be null");
        }
        this.uriReference = uriReference;
    }

    /**
     * Constructs a new <code>URIReferenceException</code> with the specified
     * cause and a detail message of <code>(cause==null ? null :
     * cause.toString())</code> (which typically contains the class and detail
     * message of <code>cause</code>).
     *
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public URIReferenceException(Throwable cause) {
        super(cause==null ? null : cause.toString());
        this.cause = cause;
    }

    /**
     * Returns the <code>URIReference</code> that was being dereferenced
     * when the exception was thrown.
     *
     * @return the <code>URIReference</code> that was being dereferenced
     * when the exception was thrown, or <code>null</code> if not specified
     */
    public URIReference getURIReference() {
        return uriReference;
    }

    /**
     * Returns the cause of this <code>URIReferenceException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>URIReferenceException</code> to get thrown.)
     *
     * @return the cause of this <code>URIReferenceException</code> or
     *    <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>URIReferenceException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
```

```
     */
    public void printStackTrace() {
        super.printStackTrace();
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>URIReferenceException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }

    /**
     * Prints this <code>URIReferenceException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        //XXX print backtrace of cause
    }
}
```

8.13 XMLCryptoContext.java

Secuencia 103: javax/xml/crypto/XMLCryptoContext.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */
/*
 * $Id: XMLCryptoContext.java,v 1.6 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

/**
 * Contains common context information for XML cryptographic operations.
 *
 * <p>This interface contains methods for setting and retrieving properties
 * that affect the processing of XML signatures or XML encrypted structures.
 *
 * <p>Note that <code>XMLCryptoContext</code> instances can contain information
 * and state specific to the XML cryptographic structure it is used with.
 * The results are unpredictable if an <code>XMLCryptoContext</code> is
 * used with multiple structures (for example, you should not use the same
 * {@link javax.xml.crypto.dsig.XMLValidateContext} instance to validate two
 * different {@link javax.xml.crypto.dsig.XMLSignature} objects).
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface XMLCryptoContext {

    /**
     * Returns the base URI.
     *
     * @return the base URI, or <code>null</code> if not specified
     * @see #setBaseURI(String)
     */
    String getBaseURI();

    /**
     * Sets the base URI.
     *
     * @param baseURI the base URI, or <code>null</code> to remove current
     *    value
     * @throws IllegalArgumentException if <code>baseURI</code> is not RFC
     *    2396 compliant
     * @see #getBaseURI
     */
    void setBaseURI(String baseURI);

    /**
     * Returns the key selector for finding a key.
     *
     * @return the key selector, or <code>null</code> if not specified
     * @see #setKeySelector(KeySelector)
     */
    KeySelector getKeySelector();
```

```

    /**
     * Sets the key selector for finding a key.
     *
     * @param ks the key selector, or <code>null</code> to remove the current
     *    setting
     * @see #getKeySelector
     */
    void setKeySelector(KeySelector ks);

    /**
     * Returns a <code>URIDereferencer</code> that is used to dereference
     * {@link URIReference}s.
     *
     * @return the <code>URIDereferencer</code>, or <code>null</code> if not
     *    specified
     * @see #setURIDereferencer(URIDereferencer)
     */
    URIDereferencer getURIDereferencer();

    /**
     * Sets a <code>URIDereferencer</code> that is used to dereference
     * {@link URIReference}s. The specified <code>URIDereferencer</code>
     * is used in place of an implementation's default
     * <code>URIDereferencer</code>.
     *
     * @param dereferencer the <code>URIDereferencer</code>, or
     *    <code>null</code> to remove any current setting
     * @see #getURIDereferencer
     */
    void setURIDereferencer(URIDereferencer dereferencer);

    /**
     * Returns the namespace prefix that the specified namespace URI is
     * associated with. Returns the specified default prefix if the specified
     * namespace URI has not been bound to a prefix. To bind a namespace URI
     * to a prefix, call the {@link #putNamespacePrefix putNamespacePrefix}
     * method.
     *
     * @param namespaceURI a namespace URI
     * @param defaultPrefix the prefix to be returned in the event that the
     *    the specified namespace URI has not been bound to a prefix.
     * @return the prefix that is associated with the specified namespace URI,
     *    or <code>defaultPrefix</code> if the URI is not registered. If
     *    the namespace URI is registered but has no prefix, an empty string
     *    (<code>""</code>) is returned.
     * @throws NullPointerException if <code>namespaceURI</code> is
     *    <code>null</code>
     * @see #putNamespacePrefix(String, String)
     */
    String getNamespacePrefix(String namespaceURI, String defaultPrefix);

    /**
```

```
     * Maps the specified namespace URI to the specified prefix. If there is
     * already a prefix associated with the specified namespace URI, the old
     * prefix is replaced by the specified prefix.
     *
     * @param namespaceURI a namespace URI
     * @param prefix a namespace prefix (or <code>null</code> to remove any
     *    existing mapping). Specifying the empty string (<code>""</code>)
     *    binds no prefix to the namespace URI.
     * @return the previous prefix associated with the specified namespace
     *    URI, or <code>null</code> if there was none
     * @throws NullPointerException if <code>namespaceURI</code> is
     *    <code>null</code>
     * @see #getNamespacePrefix(String, String)
     */
    String putNamespacePrefix(String namespaceURI, String prefix);

    /**
     * Returns the default namespace prefix. The default namespace prefix
     * is the prefix for all namespace URIs not explicitly set by the
     * {@link #putNamespacePrefix putNamespacePrefix} method.
     *
     * @return the default namespace prefix, or <code>null</code> if none has
     *    been set.
     * @see #setDefaultNamespacePrefix(String)
     */
    String getDefaultNamespacePrefix();

    /**
     * Sets the default namespace prefix. This sets the namespace prefix for
     * all namespace URIs not explicitly set by the {@link #putNamespacePrefix
     * putNamespacePrefix} method.
     *
     * @param defaultPrefix the default namespace prefix, or <code>null</code>
     *    to remove the current setting. Specify the empty string
     *    (<code>""</code>) to bind no prefix.
     * @see #getDefaultNamespacePrefix
     */
    void setDefaultNamespacePrefix(String defaultPrefix);

    /**
     * Sets the specified property.
     *
     * @param name the name of the property
     * @param value the value of the property to be set
     * @return the previous value of the specified property, or
     *    <code>null</code> if it did not have a value
     * @throws NullPointerException if <code>name</code> is <code>null</code>
     * @see #getProperty(String)
     */
    Object setProperty(String name, Object value);

    /**
     * Returns the value of the specified property.
```

```
     *
     * @param name the name of the property
     * @return the current value of the specified property, or
     *    <code>null</code> if it does not have a value
     * @throws NullPointerException if <code>name</code> is <code>null</code>
     * @see #setProperty(String, Object)
     */
    Object getProperty(String name);

    /**
     * Returns the value to which this context maps the specified key.
     *
     * <p>More formally, if this context contains a mapping from a key
     * <code>k</code> to a value <code>v</code> such that
     * <code>(key==null ? k==null : key.equals(k))</code>, then this method
     * returns <code>v</code>; otherwise it returns <code>null</code>. (There
     * can be at most one such mapping.)
     *
     * <p>This method is useful for retrieving arbitrary information that is
     * specific to the cryptographic operation that this context is used for.
     *
     * @param key the key whose associated value is to be returned
     * @return the value to which this context maps the specified key, or
     *    <code>null</code> if there is no mapping for the key
     * @see #put(Object, Object)
     */
    Object get(Object key);

    /**
     * Associates the specified value with the specified key in this context.
     * If the context previously contained a mapping for this key, the old
     * value is replaced by the specified value.
     *
     * <p>This method is useful for storing arbitrary information that is
     * specific to the cryptographic operation that this context is used for.
     *
     * @param key key with which the specified value is to be associated with
     * @param value value to be associated with the specified key
     * @return the previous value associated with the key, or <code>null</code>
     *    if there was no mapping for the key
     * @throws IllegalArgumentException if some aspect of this key or value
     *    prevents it from being stored in this context
     * @see #get(Object)
     */
    Object put(Object key, Object value);
}
```

8.14 XMLStructure.java

Secuencia 104: javax/xml/crypto/XMLStructure.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XMLStructure.java,v 1.3 2005/05/10 15:47:42 mullan Exp $
 */
package javax.xml.crypto;

/**
 * A representation of an XML structure from any namespace. The purpose of
 * this interface is to group (and provide type safety for) all
 * representations of XML structures.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface XMLStructure {

    /**
     * Indicates whether a specified feature is supported.
     *
     * @param feature the feature name (as an absolute URI)
     * @return <code>true</code> if the specified feature is supported,
     *    <code>false</code> otherwise
     * @throws NullPointerException if <code>feature</code> is <code>null</code>
     */
    boolean isFeatureSupported(String feature);
}
```

9 Xml Crypto Dom (javax.xml.crypto.dom package)

9.1 DOMCryptoContext.java

Secuencia 105: javax/xml/crypto/dom/DOMCryptoContext.java

```java
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: DOMCryptoContext.java,v 1.3 2005/05/09 18:33:26 mullan Exp $
 */
package javax.xml.crypto.dom;

import javax.xml.crypto.KeySelector;
import javax.xml.crypto.URIDereferencer;
import javax.xml.crypto.XMLCryptoContext;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import org.w3c.dom.Element;

/**
 * This class provides a DOM-specific implementation of the
 * {@link XMLCryptoContext} interface. It also includes additional
 * methods that are specific to a DOM-based implementation for registering
 * and retrieving elements that contain attributes of type ID.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public class DOMCryptoContext implements XMLCryptoContext {

    private HashMap<String,String> nsMap = new HashMap<>();
    private HashMap<String,Element> idMap = new HashMap<>();
    private HashMap<Object,Object> objMap = new HashMap<>();
    private String baseURI;
```

```
    private KeySelector ks;
    private URIDereferencer dereferencer;
    private HashMap<String,Object> propMap = new HashMap<>();
    private String defaultPrefix;

    /**
     * Default constructor. (For invocation by subclass constructors).
     */
    protected DOMCryptoContext() {}

    /**
     * This implementation uses an internal {@link HashMap} to get the prefix
     * that the specified URI maps to. It returns the <code>defaultPrefix</code>
     * if it maps to <code>null</code>.
     *
     * @throws NullPointerException {@inheritDoc}
     */
    public String getNamespacePrefix(String namespaceURI,
        String defaultPrefix) {
        if (namespaceURI == null) {
            throw new NullPointerException("namespaceURI cannot be null");
        }
        String prefix = nsMap.get(namespaceURI);
        return (prefix != null ? prefix : defaultPrefix);
    }

    /**
     * This implementation uses an internal {@link HashMap} to map the URI
     * to the specified prefix.
     *
     * @throws NullPointerException {@inheritDoc}
     */
    public String putNamespacePrefix(String namespaceURI, String prefix) {
        if (namespaceURI == null) {
            throw new NullPointerException("namespaceURI is null");
        }
        return nsMap.put(namespaceURI, prefix);
    }

    public String getDefaultNamespacePrefix() {
        return defaultPrefix;
    }

    public void setDefaultNamespacePrefix(String defaultPrefix) {
        this.defaultPrefix = defaultPrefix;
    }

    public String getBaseURI() {
        return baseURI;
    }

    /**
     * @throws IllegalArgumentException {@inheritDoc}
```

```
     */
    public void setBaseURI(String baseURI) {
        if (baseURI != null) {
            java.net.URI.create(baseURI);
        }
        this.baseURI = baseURI;
    }

    public URIDereferencer getURIDereferencer() {
        return dereferencer;
    }

    public void setURIDereferencer(URIDereferencer dereferencer) {
        this.dereferencer = dereferencer;
    }

    /**
     * This implementation uses an internal {@link HashMap} to get the object
     * that the specified name maps to.
     *
     * @throws NullPointerException {@inheritDoc}
     */
    public Object getProperty(String name) {
        if (name == null) {
            throw new NullPointerException("name is null");
        }
        return propMap.get(name);
    }

    /**
     * This implementation uses an internal {@link HashMap} to map the name
     * to the specified object.
     *
     * @throws NullPointerException {@inheritDoc}
     */
    public Object setProperty(String name, Object value) {
        if (name == null) {
            throw new NullPointerException("name is null");
        }
        return propMap.put(name, value);
    }

    public KeySelector getKeySelector() {
        return ks;
    }

    public void setKeySelector(KeySelector ks) {
        this.ks = ks;
    }

    /**
     * Returns the <code>Element</code> with the specified ID attribute value.
     *
```

```
     * <p>This implementation uses an internal {@link HashMap} to get the
     * element that the specified attribute value maps to.
     *
     * @param idValue the value of the ID
     * @return the <code>Element</code> with the specified ID attribute value,
     *    or <code>null</code> if none.
     * @throws NullPointerException if <code>idValue</code> is <code>null</code>
     * @see #setIdAttributeNS
     */
    public Element getElementById(String idValue) {
        if (idValue == null) {
            throw new NullPointerException("idValue is null");
        }
        return idMap.get(idValue);
    }

    /**
     * Registers the element's attribute specified by the namespace URI and
     * local name to be of type ID. The attribute must have a non-empty value.
     *
     * <p>This implementation uses an internal {@link HashMap} to map the
     * attribute's value to the specified element.
     *
     * @param element the element
     * @param namespaceURI the namespace URI of the attribute (specify
     *    <code>null</code> if not applicable)
     * @param localName the local name of the attribute
     * @throws IllegalArgumentException if <code>localName</code> is not an
     *    attribute of the specified element or it does not contain a specific
     *    value
     * @throws NullPointerException if <code>element</code> or
     *    <code>localName</code> is <code>null</code>
     * @see #getElementById
     */
    public void setIdAttributeNS(Element element, String namespaceURI,
        String localName) {
        if (element == null) {
            throw new NullPointerException("element is null");
        }
        if (localName == null) {
            throw new NullPointerException("localName is null");
        }
        String idValue = element.getAttributeNS(namespaceURI, localName);
        if (idValue == null || idValue.length() == 0) {
            throw new IllegalArgumentException(localName + " is not an " +
                "attribute");
        }
        idMap.put(idValue, element);
    }

    /**
     * Returns a read-only iterator over the set of Id/Element mappings of
     * this <code>DOMCryptoContext</code>. Attempts to modify the set via the
```

```
     * {@link Iterator#remove} method throw an
     * <code>UnsupportedOperationException</code>. The mappings are returned
     * in no particular order. Each element in the iteration is represented as a
     * {@link java.util.Map.Entry}. If the <code>DOMCryptoContext</code> is
     * modified while an iteration is in progress, the results of the
     * iteration are undefined.
     *
     * @return a read-only iterator over the set of mappings
     */
    @SuppressWarnings("rawtypes")
    public Iterator iterator() {
        return Collections.unmodifiableMap(idMap).entrySet().iterator();
    }

    /**
     * This implementation uses an internal {@link HashMap} to get the object
     * that the specified key maps to.
     */
    public Object get(Object key) {
        return objMap.get(key);
    }

    /**
     * This implementation uses an internal {@link HashMap} to map the key
     * to the specified object.
     *
     * @throws IllegalArgumentException {@inheritDoc}
     */
    public Object put(Object key, Object value) {
        return objMap.put(key, value);
    }
}
```

9.2 DOMStructure.java

Secuencia 106: javax/xml/crypto/dom/DOMStructure.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: DOMStructure.java,v 1.6 2005/05/09 18:33:26 mullan Exp $
 */
package javax.xml.crypto.dom;

import org.w3c.dom.Node;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.XMLSignature;

/**
 * A DOM-specific {@link XMLStructure}. The purpose of this class is to
 * allow a DOM node to be used to represent extensible content (any elements
 * or mixed content) in XML Signature structures.
 *
 * <p>If a sequence of nodes is needed, the node contained in the
 * <code>DOMStructure</code> is the first node of the sequence and successive
 * nodes can be accessed by invoking {@link Node#getNextSibling}.
 *
 * <p>If the owner document of the <code>DOMStructure</code> is different than
 * the target document of an <code>XMLSignature</code>, the
 * {@link XMLSignature#sign(XMLSignContext)} method imports the node into the
 * target document before generating the signature.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public class DOMStructure implements XMLStructure {

    private final Node node;

    /**
     * Creates a <code>DOMStructure</code> containing the specified node.
     *
     * @param node the node
     * @throws NullPointerException if <code>node</code> is <code>null</code>
     */
    public DOMStructure(Node node) {
        if (node == null) {
            throw new NullPointerException("node cannot be null");
        }
        this.node = node;
    }

    /**
     * Returns the node contained in this <code>DOMStructure</code>.
```

```
     *
     * @return the node
     */
    public Node getNode() {
        return node;
    }

    /**
     * @throws NullPointerException {@inheritDoc}
     */
    public boolean isFeatureSupported(String feature) {
        if (feature == null) {
            throw new NullPointerException();
        } else {
            return false;
        }
    }
}
```

9.3 DOMURIReference.java

Secuencia 107: javax/xml/crypto/dom/DOMURIReference.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: DOMURIReference.java,v 1.5 2005/05/09 18:33:26 mullan Exp $
 */
package javax.xml.crypto.dom;

import javax.xml.crypto.URIReference;
import org.w3c.dom.Node;
```

```java

/**
 * A DOM-specific {@link URIReference}. The purpose of this class is to
 * provide additional context necessary for resolving XPointer URIs or
 * same-document references.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface DOMURIReference extends URIReference {

    /**
     * Returns the here node.
     *
     * @return the attribute or processing instruction node or the
     *    parent element of the text node that directly contains the URI
     */
    Node getHere();
}
```

10 Xml Crypto Dsig (javax.xml.crypto.dsig package)

10.1 CanonicalizationMethod.java

Secuencia 108: javax/xml/crypto/dsig/CanonicalizationMethod.java

```java
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: CanonicalizationMethod.java,v 1.6 2005/05/10 16:03:45 mullan Exp $
 */
```

```
package javax.xml.crypto.dsig;

import java.security.spec.AlgorithmParameterSpec;
import javax.xml.crypto.dsig.spec.C14NMethodParameterSpec;

/**
 * A representation of the XML <code>CanonicalizationMethod</code>
 * element as defined in the
 * <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>. The XML
 * Schema Definition is defined as:
 * <p>
 * <pre>
 *   &lt;element name="CanonicalizationMethod" type="ds:CanonicalizationMethodType"/&gt;
 *     &lt;complexType name="CanonicalizationMethodType" mixed="true"&gt;
 *       &lt;sequence&gt;
 *         &lt;any namespace="##any" minOccurs="0" maxOccurs="unbounded"/&gt;
 *           &lt;!-- (0,unbounded) elements from (1,1) namespace --&gt;
 *       &lt;/sequence&gt;
 *       &lt;attribute name="Algorithm" type="anyURI" use="required"/&gt;
 *     &lt;/complexType&gt;
 * </pre>
 *
 * A <code>CanonicalizationMethod</code> instance may be created by invoking
 * the {@link XMLSignatureFactory#newCanonicalizationMethod
 * newCanonicalizationMethod} method of the {@link XMLSignatureFactory} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newCanonicalizationMethod(String, C14NMethodParameterSpec)
 */
public interface CanonicalizationMethod extends Transform {

    /**
     * The <a href="http://www.w3.org/TR/2001/REC-xml-c14n-20010315">Canonical
     * XML (without comments)</a> canonicalization method algorithm URI.
     */
    final static String INCLUSIVE =
        "http://www.w3.org/TR/2001/REC-xml-c14n-20010315";

    /**
     * The
     * <a href="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments">
     * Canonical XML with comments</a> canonicalization method algorithm URI.
     */
    final static String INCLUSIVE_WITH_COMMENTS =
        "http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments";

    /**
     * The <a href="http://www.w3.org/2001/10/xml-exc-c14n#">Exclusive
     * Canonical XML (without comments)</a> canonicalization method algorithm
     * URI.
```

```
     */
    final static String EXCLUSIVE =
        "http://www.w3.org/2001/10/xml-exc-c14n#";

    /**
     * The <a href="http://www.w3.org/2001/10/xml-exc-c14n#WithComments">
     * Exclusive Canonical XML with comments</a> canonicalization method
     * algorithm URI.
     */
    final static String EXCLUSIVE_WITH_COMMENTS =
        "http://www.w3.org/2001/10/xml-exc-c14n#WithComments";

    /**
     * Returns the algorithm-specific input parameters associated with this
     * <code>CanonicalizationMethod</code>.
     *
     * <p>The returned parameters can be typecast to a
     * {@link C14NMethodParameterSpec} object.
     *
     * @return the algorithm-specific input parameters (may be
     *    <code>null</code> if not specified)
     */
    AlgorithmParameterSpec getParameterSpec();
}
```

10.2 DigestMethod.java

Secuencia 109: javax/xml/crypto/dsig/DigestMethod.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
```

```
 * $Id: DigestMethod.java,v 1.6 2005/05/10 16:03:46 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.AlgorithmMethod;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.spec.DigestMethodParameterSpec;
import java.security.spec.AlgorithmParameterSpec;

/**
 * A representation of the XML <code>DigestMethod</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <p>
 * <pre>
 *   &lt;element name="DigestMethod" type="ds:DigestMethodType"/&gt;
 *     &lt;complexType name="DigestMethodType" mixed="true"&gt;
 *       &lt;sequence&gt;
 *         &lt;any namespace="##any" minOccurs="0" maxOccurs="unbounded"/&gt;
 *           &lt;!-- (0,unbounded) elements from (1,1) namespace --&gt;
 *       &lt;/sequence&gt;
 *       &lt;attribute name="Algorithm" type="anyURI" use="required"/&gt;
 *     &lt;/complexType&gt;
 * </pre>
 *
 * A <code>DigestMethod</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newDigestMethod newDigestMethod} method
 * of the {@link XMLSignatureFactory} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newDigestMethod(String, DigestMethodParameterSpec)
 */
public interface DigestMethod extends XMLStructure, AlgorithmMethod {

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#sha1">
     * SHA1</a> digest method algorithm URI.
     */
    static final String SHA1 = "http://www.w3.org/2000/09/xmldsig#sha1";

    /**
     * The <a href="http://www.w3.org/2001/04/xmlenc#sha256">
     * SHA256</a> digest method algorithm URI.
     */
    static final String SHA256 = "http://www.w3.org/2001/04/xmlenc#sha256";

    /**
     * The <a href="http://www.w3.org/2001/04/xmlenc#sha512">
     * SHA512</a> digest method algorithm URI.
     */
```

```
    static final String SHA512 = "http://www.w3.org/2001/04/xmlenc#sha512";

    /**
     * The <a href="http://www.w3.org/2001/04/xmlenc#ripemd160">
     * RIPEMD-160</a> digest method algorithm URI.
     */
    static final String RIPEMD160 = "http://www.w3.org/2001/04/xmlenc#ripemd160";

    /**
     * Returns the algorithm-specific input parameters associated with this
     * <code>DigestMethod</code>.
     *
     * <p>The returned parameters can be typecast to a {@link
     * DigestMethodParameterSpec} object.
     *
     * @return the algorithm-specific parameters (may be <code>null</code> if
     *    not specified)
     */
    AlgorithmParameterSpec getParameterSpec();
}
```

10.3 Manifest.java

Secuencia 110: javax/xml/crypto/dsig/Manifest.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: Manifest.java,v 1.7 2005/05/10 16:03:46 mullan Exp $
 */
package javax.xml.crypto.dsig;

```

```java
import javax.xml.crypto.XMLStructure;
import java.util.List;

/**
 * A representation of the XML <code>Manifest</code> element as defined in
 * the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <pre>{@code
 * <element name="Manifest" type="ds:ManifestType"/>
 *   <complexType name="ManifestType">
 *     <sequence>
 *       <element ref="ds:Reference" maxOccurs="unbounded"/>
 *     </sequence>
 *     <attribute name="Id" type="ID" use="optional"/>
 *   </complexType>
 * }</pre>
 *
 * A <code>Manifest</code> instance may be created by invoking
 * one of the {@link XMLSignatureFactory#newManifest newManifest}
 * methods of the {@link XMLSignatureFactory} class; for example:
 *
 * <pre>
 *   XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
 *   List references = Collections.singletonList(factory.newReference
 *       ("#reference-1", DigestMethod.SHA1));
 *   Manifest manifest = factory.newManifest(references, "manifest-1");
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newManifest(List)
 * @see XMLSignatureFactory#newManifest(List, String)
 */
public interface Manifest extends XMLStructure {

    /**
     * URI that identifies the <code>Manifest</code> element (this can be
     * specified as the value of the <code>type</code> parameter of the
     * {@link Reference} class to identify the referent's type).
     */
    final static String TYPE = "http://www.w3.org/2000/09/xmldsig#Manifest";

    /**
     * Returns the Id of this <code>Manifest</code>.
     *
     * @return the Id  of this <code>Manifest</code> (or <code>null</code>
     *    if not specified)
     */
    String getId();

    /**
```

```
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of one or more {@link Reference}s that are contained in this
     * <code>Manifest</code>.
     *
     * @return an unmodifiable list of one or more <code>Reference</code>s
     */
    @SuppressWarnings("rawtypes")
    List getReferences();
}
```

10.4 Reference.java

Secuencia 111: javax/xml/crypto/dsig/Reference.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: Reference.java,v 1.9 2005/05/10 16:03:46 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.Data;
import javax.xml.crypto.URIReference;
import javax.xml.crypto.XMLStructure;
import java.io.InputStream;
import java.util.List;

/**
 * A representation of the <code>Reference</code> element as defined in the
 * <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML schema is defined as:
```

```
 * <code><pre>
 * &lt;element name="Reference" type="ds:ReferenceType"/&gt;
 * &lt;complexType name="ReferenceType"&gt;
 *   &lt;sequence&gt;
 *     &lt;element ref="ds:Transforms" minOccurs="0"/&gt;
 *     &lt;element ref="ds:DigestMethod"/&gt;
 *     &lt;element ref="ds:DigestValue"/&gt;
 *   &lt;/sequence&gt;
 *   &lt;attribute name="Id" type="ID" use="optional"/&gt;
 *   &lt;attribute name="URI" type="anyURI" use="optional"/&gt;
 *   &lt;attribute name="Type" type="anyURI" use="optional"/&gt;
 * &lt;/complexType&gt;
 *
 * &lt;element name="DigestValue" type="ds:DigestValueType"/&gt;
 * &lt;simpleType name="DigestValueType"&gt;
 *   &lt;restriction base="base64Binary"/&gt;
 * &lt;/simpleType&gt;
 * </pre></code>
 *
 * <p>A <code>Reference</code> instance may be created by invoking one of the
 * {@link XMLSignatureFactory#newReference newReference} methods of the
 * {@link XMLSignatureFactory} class; for example:
 *
 * <pre>
 *   XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
 *   Reference ref = factory.newReference
 *     ("http://www.ietf.org/rfc/rfc3275.txt",
 *      factory.newDigestMethod(DigestMethod.SHA1, null));
 * </pre>
 *
 * @author Sean Mullan
 * @author Erwin van der Koogh
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newReference(String, DigestMethod)
 * @see XMLSignatureFactory#newReference(String, DigestMethod, List, String, String)
 */
public interface Reference extends URIReference, XMLStructure {

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of {@link Transform}s that are contained in this
     * <code>Reference</code>.
     *
     * @return an unmodifiable list of <code>Transform</code>s
     *    (may be empty but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    List getTransforms();

    /**
     * Returns the digest method of this <code>Reference</code>.
     *
```

```
     * @return the digest method
     */
    DigestMethod getDigestMethod();

    /**
     * Returns the optional <code>Id</code> attribute of this
     * <code>Reference</code>, which permits this reference to be
     * referenced from elsewhere.
     *
     * @return the <code>Id</code> attribute (may be <code>null</code> if not
     *    specified)
     */
    String getId();

    /**
     * Returns the digest value of this <code>Reference</code>.
     *
     * @return the raw digest value, or <code>null</code> if this reference has
     *    not been digested yet. Each invocation of this method returns a new
     *    clone to protect against subsequent modification.
     */
    byte[] getDigestValue();

    /**
     * Returns the calculated digest value of this <code>Reference</code>
     * after a validation operation. This method is useful for debugging if
     * the reference fails to validate.
     *
     * @return the calculated digest value, or <code>null</code> if this
     *    reference has not been validated yet. Each invocation of this method
     *    returns a new clone to protect against subsequent modification.
     */
    byte[] getCalculatedDigestValue();

    /**
     * Validates this reference. This method verifies the digest of this
     * reference.
     *
     * <p>This method only validates the reference the first time it is
     * invoked. On subsequent invocations, it returns a cached result.
     *
     * @return <code>true</code> if this reference was validated successfully;
     *    <code>false</code> otherwise
     * @param validateContext the validating context
     * @throws NullPointerException if <code>validateContext</code> is
     *    <code>null</code>
     * @throws XMLSignatureException if an unexpected exception occurs while
     *    validating the reference
     */
    boolean validate(XMLValidateContext validateContext)
        throws XMLSignatureException;

    /**
```

```
     * Returns the dereferenced data, if
     * <a href="XMLSignContext.html#Supported Properties">reference caching</a>
     * is enabled. This is the result of dereferencing the URI of this
     * reference during a validation or generation operation.
     *
     * @return the dereferenced data, or <code>null</code> if reference
     *    caching is not enabled or this reference has not been generated or
     *    validated
     */
    Data getDereferencedData();

    /**
     * Returns the pre-digested input stream, if
     * <a href="XMLSignContext.html#Supported Properties">reference caching</a>
     * is enabled. This is the input to the digest operation during a
     * validation or signing operation.
     *
     * @return an input stream containing the pre-digested input, or
     *    <code>null</code> if reference caching is not enabled or this
     *    reference has not been generated or validated
     */
    InputStream getDigestInputStream();
}
```

10.5 SignatureMethod.java

Secuencia 112: javax/xml/crypto/dsig/SignatureMethod.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: SignatureMethod.java,v 1.5 2005/05/10 16:03:46 mullan Exp $
```

```
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.AlgorithmMethod;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.spec.SignatureMethodParameterSpec;
import java.security.spec.AlgorithmParameterSpec;

/**
 * A representation of the XML <code>SignatureMethod</code> element
 * as defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <p>
 * <pre>
 *   &lt;element name="SignatureMethod" type="ds:SignatureMethodType"/&gt;
 *     &lt;complexType name="SignatureMethodType" mixed="true"&gt;
 *       &lt;sequence&gt;
 *         &lt;element name="HMACOutputLength" minOccurs="0" type="ds:HMACOutputLengthType"/&gt;
 *         &lt;any namespace="##any" minOccurs="0" maxOccurs="unbounded"/&gt;
 *           &lt;!-- (0,unbounded) elements from (1,1) namespace --&gt;
 *       &lt;/sequence&gt;
 *       &lt;attribute name="Algorithm" type="anyURI" use="required"/&gt;
 *     &lt;/complexType&gt;
 * </pre>
 *
 * A <code>SignatureMethod</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newSignatureMethod newSignatureMethod} method
 * of the {@link XMLSignatureFactory} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newSignatureMethod(String, SignatureMethodParameterSpec)
 */
public interface SignatureMethod extends XMLStructure, AlgorithmMethod {

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#dsa-sha1">DSAwithSHA1</a>
     * (DSS) signature method algorithm URI.
     */
    static final String DSA_SHA1 =
        "http://www.w3.org/2000/09/xmldsig#dsa-sha1";

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#rsa-sha1">RSAwithSHA1</a>
     * (PKCS #1) signature method algorithm URI.
     */
    static final String RSA_SHA1 =
        "http://www.w3.org/2000/09/xmldsig#rsa-sha1";

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#hmac-sha1">HMAC-SHA1</a>
```

```
     * MAC signature method algorithm URI
     */
    static final String HMAC_SHA1 =
        "http://www.w3.org/2000/09/xmldsig#hmac-sha1";

    /**
     * Returns the algorithm-specific input parameters of this
     * <code>SignatureMethod</code>.
     *
     * <p>The returned parameters can be typecast to a {@link
     * SignatureMethodParameterSpec} object.
     *
     * @return the algorithm-specific input parameters of this
     *    <code>SignatureMethod</code> (may be <code>null</code> if not
     *    specified)
     */
    AlgorithmParameterSpec getParameterSpec();
}
```

10.6 SignatureProperties.java

Secuencia 113: javax/xml/crypto/dsig/SignatureProperties.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: SignatureProperties.java,v 1.4 2005/05/10 16:03:46 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.XMLStructure;
import java.util.List;
```

```java

/**
 * A representation of the XML <code>SignatureProperties</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <pre><code>
 *&lt;element name="SignatureProperties" type="ds:SignaturePropertiesType"/&gt;
 *   &lt;complexType name="SignaturePropertiesType"&gt;
 *     &lt;sequence&gt;
 *       &lt;element ref="ds:SignatureProperty" maxOccurs="unbounded"/&gt;
 *     &lt;/sequence&gt;
 *     &lt;attribute name="Id" type="ID" use="optional"/&gt;
 *   &lt;/complexType&gt;
 * </code></pre>
 *
 * A <code>SignatureProperties</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newSignatureProperties newSignatureProperties}
 * method of the {@link XMLSignatureFactory} class; for example:
 *
 * <pre>
 *   XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
 *   SignatureProperties properties =
 *      factory.newSignatureProperties(props, "signature-properties-1");
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newSignatureProperties(List, String)
 * @see SignatureProperty
 */
public interface SignatureProperties extends XMLStructure {

    /**
     * URI that identifies the <code>SignatureProperties</code> element (this
     * can be specified as the value of the <code>type</code> parameter of the
     * {@link Reference} class to identify the referent's type).
     */
    final static String TYPE =
        "http://www.w3.org/2000/09/xmldsig#SignatureProperties";

    /**
     * Returns the Id of this <code>SignatureProperties</code>.
     *
     * @return the Id of this <code>SignatureProperties</code> (or
     *    <code>null</code> if not specified)
     */
    String getId();

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of one or more {@link SignatureProperty}s that are contained in
```

```
     * this <code>SignatureProperties</code>.
     *
     * @return an unmodifiable list of one or more
     *    <code>SignatureProperty</code>s
     */
    @SuppressWarnings("rawtypes")
    List getProperties();
}
```

10.7 SignatureProperty.java

Secuencia 114: javax/xml/crypto/dsig/SignatureProperty.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: SignatureProperty.java,v 1.4 2005/05/10 16:03:46 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.XMLStructure;
import java.util.List;

/**
 * A representation of the XML <code>SignatureProperty</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <pre><code>
 *&lt;element name="SignatureProperty" type="ds:SignaturePropertyType"/&gt;
 *   &lt;complexType name="SignaturePropertyType" mixed="true"&gt;
 *     &lt;choice maxOccurs="unbounded"&gt;
```

```
 *       &lt;any namespace="##other" processContents="lax"/&gt;
 *       &lt;!-- (1,1) elements from (1, unbounded) namespaces --&gt;
 *     &lt;/choice&gt;
 *     &lt;attribute name="Target" type="anyURI" use="required"/&gt;
 *     &lt;attribute name="Id" type="ID" use="optional"/&gt;
 *   &lt;/complexType&gt;
 * </code></pre>
 *
 * A <code>SignatureProperty</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newSignatureProperty newSignatureProperty}
 * method of the {@link XMLSignatureFactory} class; for example:
 *
 * <pre>
 *   XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
 *   SignatureProperty property = factory.newSignatureProperty
 *      (Collections.singletonList(content), "#Signature-1", "TimeStamp");
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newSignatureProperty(List, String, String)
 * @see SignatureProperties
 */
public interface SignatureProperty extends XMLStructure {

    /**
     * Returns the target URI of this <code>SignatureProperty</code>.
     *
     * @return the target URI of this <code>SignatureProperty</code> (never
     *    <code>null</code>)
     */
    String getTarget();

    /**
     * Returns the Id of this <code>SignatureProperty</code>.
     *
     * @return the Id of this <code>SignatureProperty</code> (or
     *    <code>null</code> if not specified)
     */
    String getId();

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of one or more {@link XMLStructure}s that are contained in
     * this <code>SignatureProperty</code>. These represent additional
     * information items concerning the generation of the {@link XMLSignature}
     * (i.e. date/time stamp or serial numbers of cryptographic hardware used
     * in signature generation).
     *
     * @return an unmodifiable list of one or more <code>XMLStructure</code>s
     */
    @SuppressWarnings("rawtypes")
```

```
    List getContent();
}
```

10.8 SignedInfo.java

Secuencia 115: javax/xml/crypto/dsig/SignedInfo.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: SignedInfo.java,v 1.7 2005/05/10 16:03:47 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.XMLStructure;
import java.io.InputStream;
import java.util.List;

/**
 * An representation of the XML <code>SignedInfo</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 * <pre><code>
 * &lt;element name="SignedInfo" type="ds:SignedInfoType"/&gt;
 * &lt;complexType name="SignedInfoType"&gt;
 *   &lt;sequence&gt;
 *     &lt;element ref="ds:CanonicalizationMethod"/&gt;
 *     &lt;element ref="ds:SignatureMethod"/&gt;
 *     &lt;element ref="ds:Reference" maxOccurs="unbounded"/&gt;
 *   &lt;/sequence&gt;
 *   &lt;attribute name="Id" type="ID" use="optional"/&gt;
```

```
 * &lt;/complexType&gt;
 * </code></pre>
 *
 * A <code>SignedInfo</code> instance may be created by invoking one of the
 * {@link XMLSignatureFactory#newSignedInfo newSignedInfo} methods of the
 * {@link XMLSignatureFactory} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newSignedInfo(CanonicalizationMethod, SignatureMethod, List)
 * @see XMLSignatureFactory#newSignedInfo(CanonicalizationMethod, SignatureMethod, List, String)
 */
public interface SignedInfo extends XMLStructure {

    /**
     * Returns the canonicalization method of this <code>SignedInfo</code>.
     *
     * @return the canonicalization method
     */
    CanonicalizationMethod getCanonicalizationMethod();

    /**
     * Returns the signature method of this <code>SignedInfo</code>.
     *
     * @return the signature method
     */
    SignatureMethod getSignatureMethod();

    /**
     * Returns an {@link java.util.Collections#unmodifiableList
     * unmodifiable list} of one or more {@link Reference}s.
     *
     * @return an unmodifiable list of one or more {@link Reference}s
     */
    @SuppressWarnings("rawtypes")
    List getReferences();

    /**
     * Returns the optional <code>Id</code> attribute of this
     * <code>SignedInfo</code>.
     *
     * @return the id (may be <code>null</code> if not specified)
     */
    String getId();

    /**
     * Returns the canonicalized signed info bytes after a signing or
     * validation operation. This method is useful for debugging.
     *
     * @return an <code>InputStream</code> containing the canonicalized bytes,
     *    or <code>null</code> if this <code>SignedInfo</code> has not been
     *    signed or validated yet
```

```
     */
    InputStream getCanonicalizedData();
}
```

10.9 Transform.java

Secuencia 116: javax/xml/crypto/dsig/Transform.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: Transform.java,v 1.5 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;

import java.io.OutputStream;
import java.security.spec.AlgorithmParameterSpec;
import javax.xml.crypto.AlgorithmMethod;
import javax.xml.crypto.Data;
import javax.xml.crypto.OctetStreamData;
import javax.xml.crypto.XMLCryptoContext;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.spec.TransformParameterSpec;

/**
 * A representation of the XML <code>Transform</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * The XML Schema Definition is defined as:
 *
 * <pre>
 * &lt;element name="Transform" type="ds:TransformType"/&gt;
```

```
 *   &lt;complexType name="TransformType" mixed="true"&gt;
 *     &lt;choice minOccurs="0" maxOccurs="unbounded"&gt;
 *       &lt;any namespace="##other" processContents="lax"/&gt;
 *       &lt;!-- (1,1) elements from (0,unbounded) namespaces --&gt;
 *       &lt;element name="XPath" type="string"/&gt;
 *     &lt;/choice&gt;
 *     &lt;attribute name="Algorithm" type="anyURI" use="required"/&gt;
 *   &lt;/complexType&gt;
 * </pre>
 *
 * A <code>Transform</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newTransform newTransform} method
 * of the {@link XMLSignatureFactory} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#newTransform(String, TransformParameterSpec)
 */
public interface Transform extends XMLStructure, AlgorithmMethod {

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#base64">Base64</a>
     * transform algorithm URI.
     */
    final static String BASE64 = "http://www.w3.org/2000/09/xmldsig#base64";

    /**
     * The <a href="http://www.w3.org/2000/09/xmldsig#enveloped-signature">
     * Enveloped Signature</a> transform algorithm URI.
     */
    final static String ENVELOPED =
        "http://www.w3.org/2000/09/xmldsig#enveloped-signature";

    /**
     * The <a href="http://www.w3.org/TR/1999/REC-xpath-19991116">XPath</a>
     * transform algorithm URI.
     */
    final static String XPATH = "http://www.w3.org/TR/1999/REC-xpath-19991116";

    /**
     * The <a href="http://www.w3.org/2002/06/xmldsig-filter2">
     * XPath Filter 2</a> transform algorithm URI.
     */
    final static String XPATH2 = "http://www.w3.org/2002/06/xmldsig-filter2";

    /**
     * The <a href="http://www.w3.org/TR/1999/REC-xslt-19991116">XSLT</a>
     * transform algorithm URI.
     */
    final static String XSLT = "http://www.w3.org/TR/1999/REC-xslt-19991116";

    /**
```

```
     * Returns the algorithm-specific input parameters associated with this
     * <code>Transform</code>.
     * <p>
     * The returned parameters can be typecast to a
     * {@link TransformParameterSpec} object.
     *
     * @return the algorithm-specific input parameters (may be <code>null</code>
     *    if not specified)
     */
    AlgorithmParameterSpec getParameterSpec();

    /**
     * Transforms the specified data using the underlying transform algorithm.
     *
     * @param data the data to be transformed
     * @param context the <code>XMLCryptoContext</code> containing
     *    additional context (may be <code>null</code> if not applicable)
     * @return the transformed data
     * @throws NullPointerException if <code>data</code> is <code>null</code>
     * @throws TransformException if an error occurs while executing the
     *    transform
     */
    public abstract Data transform(Data data, XMLCryptoContext context)
        throws TransformException;

    /**
     * Transforms the specified data using the underlying transform algorithm.
     * If the output of this transform is an <code>OctetStreamData</code>, then
     * this method returns <code>null</code> and the bytes are written to the
     * specified <code>OutputStream</code>. Otherwise, the
     * <code>OutputStream</code> is ignored and the method behaves as if
     * {@link #transform(Data, XMLCryptoContext)} were invoked.
     *
     * @param data the data to be transformed
     * @param context the <code>XMLCryptoContext</code> containing
     *    additional context (may be <code>null</code> if not applicable)
     * @param os the <code>OutputStream</code> that should be used to write
     *    the transformed data to
     * @return the transformed data (or <code>null</code> if the data was
     *    written to the <code>OutputStream</code> parameter)
     * @throws NullPointerException if <code>data</code> or <code>os</code>
     *    is <code>null</code>
     * @throws TransformException if an error occurs while executing the
     *    transform
     */
    public abstract Data transform
        (Data data, XMLCryptoContext context, OutputStream os)
        throws TransformException;
}
```

10.10 TransformException.java

Secuencia 117: javax/xml/crypto/dsig/TransformException.java

```java
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: TransformException.java,v 1.3 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;

import java.io.PrintStream;
import java.io.PrintWriter;

/**
 * Indicates an exceptional condition that occurred while executing a
 * transform algorithm.
 *
 * <p>A <code>TransformException</code> can contain a cause: another
 * throwable that caused this <code>TransformException</code> to get thrown.
 *
 * @see Transform#transform
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public class TransformException extends Exception {

    private static final long serialVersionUID = 5082634801360427800L;

    /**
     * The throwable that caused this exception to get thrown, or null if this
     * exception was not caused by another throwable or if the causative
     * throwable is unknown.
     *
```

```
     * @serial
     */
    private Throwable cause;

    /**
     * Constructs a new <code>TransformException</code> with
     * <code>null</code> as its detail message.
     */
    public TransformException() {
        super();
    }

    /**
     * Constructs a new <code>TransformException</code> with the specified
     * detail message.
     *
     * @param message the detail message
     */
    public TransformException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>TransformException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public TransformException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>TransformException</code> with the specified
     * cause and a detail message of
     * <code>(cause==null ? null : cause.toString())</code>
     * (which typically contains the class and detail message of
     * <code>cause</code>).
     *
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public TransformException(Throwable cause) {
        super(cause==null ? null : cause.toString());
        this.cause = cause;
    }

```

```
    /**
     * Returns the cause of this <code>TransformException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>TransformException</code> to get thrown.)
     *
     * @return the cause of this <code>TransformException</code> or
     *         <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>TransformException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
     */
    public void printStackTrace() {
        super.printStackTrace();
        if (cause != null) {
            cause.printStackTrace();
        }
    }

    /**
     * Prints this <code>TransformException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        if (cause != null) {
            cause.printStackTrace(s);
        }
    }

    /**
     * Prints this <code>TransformException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        if (cause != null) {
            cause.printStackTrace(s);
        }
    }
}
```

10.11 TransformService.java

Secuencia 118: javax/xml/crypto/dsig/TransformService.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: TransformService.java,v 1.6.4.1 2005/09/15 12:42:11 mullan Exp $
 */
package javax.xml.crypto.dsig;

import java.security.InvalidAlgorithmParameterException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Provider;
import java.security.Provider.Service;
import java.security.Security;
import java.util.*;
import javax.xml.crypto.MarshalException;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.XMLCryptoContext;
import javax.xml.crypto.dsig.spec.TransformParameterSpec;

import sun.security.jca.*;
import sun.security.jca.GetInstance.Instance;

/**
 * A Service Provider Interface for transform and canonicalization algorithms.
 *
 * <p>Each instance of <code>TransformService</code> supports a specific
 * transform or canonicalization algorithm and XML mechanism type. To create a
 * <code>TransformService</code>, call one of the static
 * {@link #getInstance getInstance} methods, passing in the algorithm URI and
 * XML mechanism type desired, for example:
```

```
*
* <blockquote><code>
* TransformService ts = TransformService.getInstance(Transform.XPATH2, "DOM");
* </code></blockquote>
*
* <p><code>TransformService</code> implementations are registered and loaded
* using the {@link java.security.Provider} mechanism.  Each
* <code>TransformService</code> service provider implementation should include
* a <code>MechanismType</code> service attribute that identifies the XML
* mechanism type that it supports. If the attribute is not specified,
* "DOM" is assumed. For example, a service provider that supports the
* XPath Filter 2 Transform and DOM mechanism would be specified in the
* <code>Provider</code> subclass as:
* <pre>
*     put("TransformService." + Transform.XPATH2,
*         "org.example.XPath2TransformService");
*     put("TransformService." + Transform.XPATH2 + " MechanismType", "DOM");
* </pre>
* <code>TransformService</code> implementations that support the DOM
* mechanism type must abide by the DOM interoperability requirements defined
* in the
* <a href="../../../../../technotes/guides/security/xmldsig/overview.html#DOM Mechanism
    Requirements">
* DOM Mechanism Requirements</a> section of the API overview. See the
* <a href="../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
* Service Providers</a> section of the API overview for a list of standard
* mechanism types.
* <p>
* Once a <code>TransformService</code> has been created, it can be used
* to process <code>Transform</code> or <code>CanonicalizationMethod</code>
* objects. If the <code>Transform</code> or <code>CanonicalizationMethod</code>
* exists in XML form (for example, when validating an existing
* <code>XMLSignature</code>), the {@link #init(XMLStructure, XMLCryptoContext)}
* method must be first called to initialize the transform and provide document
* context (even if there are no parameters). Alternatively, if the
* <code>Transform</code> or <code>CanonicalizationMethod</code> is being
* created from scratch, the {@link #init(TransformParameterSpec)} method
* is called to initialize the transform with parameters and the
* {@link #marshalParams marshalParams} method is called to marshal the
* parameters to XML and provide the transform with document context. Finally,
* the {@link #transform transform} method is called to perform the
* transformation.
* <p>
* <b>Concurrent Access</b>
* <p>The static methods of this class are guaranteed to be thread-safe.
* Multiple threads may concurrently invoke the static methods defined in this
* class with no ill effects.
*
* <p>However, this is not true for the non-static methods defined by this
* class. Unless otherwise documented by a specific provider, threads that
* need to access a single <code>TransformService</code> instance
* concurrently should synchronize amongst themselves and provide the
* necessary locking. Multiple threads each manipulating a different
```

```
 * <code>TransformService</code> instance need not synchronize.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public abstract class TransformService implements Transform {

    private String algorithm;
    private String mechanism;
    private Provider provider;

    /**
     * Default constructor, for invocation by subclasses.
     */
    protected TransformService() {}

    /**
     * Returns a <code>TransformService</code> that supports the specified
     * algorithm URI (ex: {@link Transform#XPATH2}) and mechanism type
     * (ex: DOM).
     *
     * <p>This method uses the standard JCA provider lookup mechanism to
     * locate and instantiate a <code>TransformService</code> implementation
     * of the desired algorithm and <code>MechanismType</code> service
     * attribute. It traverses the list of registered security
     * <code>Provider</code>s, starting with the most preferred
     * <code>Provider</code>. A new <code>TransformService</code> object
     * from the first <code>Provider</code> that supports the specified
     * algorithm and mechanism type is returned.
     *
     * <p> Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param algorithm the URI of the algorithm
     * @param mechanismType the type of the XML processing mechanism and
     *   representation
     * @return a new <code>TransformService</code>
     * @throws NullPointerException if <code>algorithm</code> or
     *   <code>mechanismType</code> is  <code>null</code>
     * @throws NoSuchAlgorithmException if no <code>Provider</code> supports a
     *   <code>TransformService</code> implementation for the specified
     *   algorithm and mechanism type
     * @see Provider
     */
    public static TransformService getInstance
        (String algorithm, String mechanismType)
        throws NoSuchAlgorithmException {
        if (mechanismType == null || algorithm == null) {
            throw new NullPointerException();
        }
        boolean dom = false;
        if (mechanismType.equals("DOM")) {
```

```
            dom = true;
        }
        List<Service> services = GetInstance.getServices("TransformService", algorithm);
        for (Iterator<Service> t = services.iterator(); t.hasNext(); ) {
            Service s = t.next();
            String value = s.getAttribute("MechanismType");
            if ((value == null && dom) ||
                (value != null && value.equals(mechanismType))) {
                Instance instance = GetInstance.getInstance(s, null);
                TransformService ts = (TransformService) instance.impl;
                ts.algorithm = algorithm;
                ts.mechanism = mechanismType;
                ts.provider = instance.provider;
                return ts;
            }
        }
        throw new NoSuchAlgorithmException
            (algorithm + " algorithm and " + mechanismType
                 + " mechanism not available");
    }

    /**
     * Returns a <code>TransformService</code> that supports the specified
     * algorithm URI (ex: {@link Transform#XPATH2}) and mechanism type
     * (ex: DOM) as supplied by the specified provider. Note that the specified
     * <code>Provider</code> object does not have to be registered in the
     * provider list.
     *
     * @param algorithm the URI of the algorithm
     * @param mechanismType the type of the XML processing mechanism and
     *   representation
     * @param provider the <code>Provider</code> object
     * @return a new <code>TransformService</code>
     * @throws NullPointerException if <code>provider</code>,
     *   <code>algorithm</code>, or <code>mechanismType</code> is
     *   <code>null</code>
     * @throws NoSuchAlgorithmException if a <code>TransformService</code>
     *   implementation for the specified algorithm and mechanism type is not
     *   available from the specified <code>Provider</code> object
     * @see Provider
     */
    public static TransformService getInstance
        (String algorithm, String mechanismType, Provider provider)
        throws NoSuchAlgorithmException {
        if (mechanismType == null || algorithm == null || provider == null) {
            throw new NullPointerException();
        }

        boolean dom = false;
        if (mechanismType.equals("DOM")) {
            dom = true;
        }
        Service s = GetInstance.getService
```

```
            ("TransformService", algorithm, provider);
        String value = s.getAttribute("MechanismType");
        if ((value == null && dom) ||
            (value != null && value.equals(mechanismType))) {
            Instance instance = GetInstance.getInstance(s, null);
            TransformService ts = (TransformService) instance.impl;
            ts.algorithm = algorithm;
            ts.mechanism = mechanismType;
            ts.provider = instance.provider;
            return ts;
        }
        throw new NoSuchAlgorithmException
            (algorithm + " algorithm and " + mechanismType
                 + " mechanism not available");
    }

    /**
     * Returns a <code>TransformService</code> that supports the specified
     * algorithm URI (ex: {@link Transform#XPATH2}) and mechanism type
     * (ex: DOM) as supplied by the specified provider. The specified provider
     * must be registered in the security provider list.
     *
     * <p>Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param algorithm the URI of the algorithm
     * @param mechanismType the type of the XML processing mechanism and
     *   representation
     * @param provider the string name of the provider
     * @return a new <code>TransformService</code>
     * @throws NoSuchProviderException if the specified provider is not
     *   registered in the security provider list
     * @throws NullPointerException if <code>provider</code>,
     *   <code>mechanismType</code>, or <code>algorithm</code> is
     *   <code>null</code>
     * @throws NoSuchAlgorithmException if a <code>TransformService</code>
     *   implementation for the specified algorithm and mechanism type is not
     *   available from the specified provider
     * @see Provider
     */
    public static TransformService getInstance
        (String algorithm, String mechanismType, String provider)
        throws NoSuchAlgorithmException, NoSuchProviderException {
        if (mechanismType == null || algorithm == null || provider == null) {
            throw new NullPointerException();
        } else if (provider.length() == 0) {
            throw new NoSuchProviderException();
        }
        boolean dom = false;
        if (mechanismType.equals("DOM")) {
            dom = true;
        }
        Service s = GetInstance.getService
```

```
            ("TransformService", algorithm, provider);
        String value = s.getAttribute("MechanismType");
        if ((value == null && dom) ||
            (value != null && value.equals(mechanismType))) {
            Instance instance = GetInstance.getInstance(s, null);
            TransformService ts = (TransformService) instance.impl;
            ts.algorithm = algorithm;
            ts.mechanism = mechanismType;
            ts.provider = instance.provider;
            return ts;
        }
        throw new NoSuchAlgorithmException
            (algorithm + " algorithm and " + mechanismType
                 + " mechanism not available");
    }

    private static class MechanismMapEntry implements Map.Entry<String,String> {
        private final String mechanism;
        private final String algorithm;
        private final String key;
        MechanismMapEntry(String algorithm, String mechanism) {
            this.algorithm = algorithm;
            this.mechanism = mechanism;
            this.key = "TransformService." + algorithm + " MechanismType";
        }
        public boolean equals(Object o) {
            if (!(o instanceof Map.Entry)) {
                return false;
            }
            Map.Entry<?,?> e = (Map.Entry<?,?>) o;
            return (getKey()==null ?
                    e.getKey()==null : getKey().equals(e.getKey())) &&
                   (getValue()==null ?
                    e.getValue()==null : getValue().equals(e.getValue()));
        }
        public String getKey() {
            return key;
        }
        public String getValue() {
            return mechanism;
        }
        public String setValue(String value) {
            throw new UnsupportedOperationException();
        }
        public int hashCode() {
            return (getKey()==null ? 0 : getKey().hashCode()) ^
                   (getValue()==null ? 0 : getValue().hashCode());
        }
    }

    /**
     * Returns the mechanism type supported by this <code>TransformService</code>.
     *
```

```
     * @return the mechanism type
     */
    public final String getMechanismType() {
        return mechanism;
    }

    /**
     * Returns the URI of the algorithm supported by this
     * <code>TransformService</code>.
     *
     * @return the algorithm URI
     */
    public final String getAlgorithm() {
        return algorithm;
    }

    /**
     * Returns the provider of this <code>TransformService</code>.
     *
     * @return the provider
     */
    public final Provider getProvider() {
        return provider;
    }

    /**
     * Initializes this <code>TransformService</code> with the specified
     * parameters.
     *
     * <p>If the parameters exist in XML form, the
     * {@link #init(XMLStructure, XMLCryptoContext)} method should be used to
     * initialize the <code>TransformService</code>.
     *
     * @param params the algorithm parameters (may be <code>null</code> if
     *   not required or optional)
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *   are invalid for this algorithm
     */
    public abstract void init(TransformParameterSpec params)
        throws InvalidAlgorithmParameterException;

    /**
     * Marshals the algorithm-specific parameters. If there are no parameters
     * to be marshalled, this method returns without throwing an exception.
     *
     * @param parent a mechanism-specific structure containing the parent
     *    node that the marshalled parameters should be appended to
     * @param context the <code>XMLCryptoContext</code> containing
     *    additional context (may be <code>null</code> if not applicable)
     * @throws ClassCastException if the type of <code>parent</code> or
     *    <code>context</code> is not compatible with this
     *    <code>TransformService</code>
     * @throws NullPointerException if <code>parent</code> is <code>null</code>
```

```
     * @throws MarshalException if the parameters cannot be marshalled
     */
    public abstract void marshalParams
        (XMLStructure parent, XMLCryptoContext context)
        throws MarshalException;

    /**
     * Initializes this <code>TransformService</code> with the specified
     * parameters and document context.
     *
     * @param parent a mechanism-specific structure containing the parent
     *    structure
     * @param context the <code>XMLCryptoContext</code> containing
     *    additional context (may be <code>null</code> if not applicable)
     * @throws ClassCastException if the type of <code>parent</code> or
     *    <code>context</code> is not compatible with this
     *    <code>TransformService</code>
     * @throws NullPointerException if <code>parent</code> is <code>null</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *   are invalid for this algorithm
     */
    public abstract void init(XMLStructure parent, XMLCryptoContext context)
        throws InvalidAlgorithmParameterException;
}
```

10.12 XMLObject.java

Secuencia 119: javax/xml/crypto/dsig/XMLObject.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```
/*
 * ===========================================================================
 *
 * (C) Copyright IBM Corp. 2003 All Rights Reserved.
 *
 * ===========================================================================
 */
/*
 * $Id: XMLObject.java,v 1.5 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;

import java.util.List;
import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>Object</code> element as defined in
 * the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * An <code>XMLObject</code> may contain any data and may include optional
 * MIME type, ID, and encoding attributes. The XML Schema Definition is
 * defined as:
 *
 * <pre><code>
 * &lt;element name="Object" type="ds:ObjectType"/&gt;
 * &lt;complexType name="ObjectType" mixed="true"&gt;
 *   &lt;sequence minOccurs="0" maxOccurs="unbounded"&gt;
 *     &lt;any namespace="##any" processContents="lax"/&gt;
 *   &lt;/sequence&gt;
 *   &lt;attribute name="Id" type="ID" use="optional"/&gt;
 *   &lt;attribute name="MimeType" type="string" use="optional"/&gt;
 *   &lt;attribute name="Encoding" type="anyURI" use="optional"/&gt;
 * &lt;/complexType&gt;
 * </code></pre>
 *
 * A <code>XMLObject</code> instance may be created by invoking the
 * {@link XMLSignatureFactory#newXMLObject newXMLObject} method of the
 * {@link XMLSignatureFactory} class; for example:
 *
 * <pre>
 *   XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
 *   List content = Collections.singletonList(fac.newManifest(references)));
 *   XMLObject object = factory.newXMLObject(content, "object-1", null, null);
 * </pre>
 *
 * <p>Note that this class is named <code>XMLObject</code> rather than
 * <code>Object</code> to avoid naming clashes with the existing
 * {@link java.lang.Object java.lang.Object} class.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @author Joyce L. Leung
 * @since 1.6
```

```
 * @see XMLSignatureFactory#newXMLObject(List, String, String, String)
 */
public interface XMLObject extends XMLStructure {

    /**
     * URI that identifies the <code>Object</code> element (this can be
     * specified as the value of the <code>type</code> parameter of the
     * {@link Reference} class to identify the referent's type).
     */
    final static String TYPE = "http://www.w3.org/2000/09/xmldsig#Object";

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of {@link XMLStructure}s contained in this <code>XMLObject</code>,
     * which represent elements from any namespace.
     *
     *<p>If there is a public subclass representing the type of
     * <code>XMLStructure</code>, it is returned as an instance of that class
     * (ex: a <code>SignatureProperties</code> element would be returned
     * as an instance of {@link javax.xml.crypto.dsig.SignatureProperties}).
     *
     * @return an unmodifiable list of <code>XMLStructure</code>s (may be empty
     *    but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    List getContent();

    /**
     * Returns the Id of this <code>XMLObject</code>.
     *
     * @return the Id (or <code>null</code> if not specified)
     */
    String getId();

    /**
     * Returns the mime type of this <code>XMLObject</code>. The
     * mime type is an optional attribute which describes the data within this
     * <code>XMLObject</code> (independent of its encoding).
     *
     * @return the mime type (or <code>null</code> if not specified)
     */
    String getMimeType();

    /**
     * Returns the encoding URI of this <code>XMLObject</code>. The encoding
     * URI identifies the method by which the object is encoded.
     *
     * @return the encoding URI (or <code>null</code> if not specified)
     */
    String getEncoding();
}
```

10.13 XMLSignContext.java

Secuencia 120: javax/xml/crypto/dsig/XMLSignContext.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XMLSignContext.java,v 1.8 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.KeySelector;
import javax.xml.crypto.XMLCryptoContext;

/**
 * Contains context information for generating XML Signatures. This interface
 * is primarily intended for type-safety.
 *
 * <p>Note that <code>XMLSignContext</code> instances can contain
 * information and state specific to the XML signature structure it is
 * used with. The results are unpredictable if an
 * <code>XMLSignContext</code> is used with different signature structures
 * (for example, you should not use the same <code>XMLSignContext</code>
 * instance to sign two different {@link XMLSignature} objects).
 * <p>
 * <b><a name="Supported Properties"></a>Supported Properties</b>
 * <p>The following properties can be set using the
 * {@link #setProperty setProperty} method.
 * <ul>
 *   <li><code>javax.xml.crypto.dsig.cacheReference</code>: value must be a
 *      {@link Boolean}. This property controls whether or not the digested
 *      {@link Reference} objects will cache the dereferenced content and
```

```
 *      pre-digested input for subsequent retrieval via the
 *      {@link Reference#getDereferencedData Reference.getDereferencedData} and
 *      {@link Reference#getDigestInputStream Reference.getDigestInputStream}
 *      methods. The default value if not specified is
 *      <code>Boolean.FALSE</code>.
 * </ul>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignature#sign(XMLSignContext)
 */
public interface XMLSignContext extends XMLCryptoContext {}
```

10.14 XMLSignature.java

Secuencia 121: javax/xml/crypto/dsig/XMLSignature.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * ============================================================================
 *
 * (C) Copyright IBM Corp. 2003 All Rights Reserved.
 *
 * ============================================================================
 */
/*
 * $Id: XMLSignature.java,v 1.10 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;
```

```

import javax.xml.crypto.KeySelector;
import javax.xml.crypto.KeySelectorResult;
import javax.xml.crypto.MarshalException;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.keyinfo.KeyInfo;
import java.security.Signature;
import java.util.List;

/**
 * A representation of the XML <code>Signature</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * This class contains methods for signing and validating XML signatures
 * with behavior as defined by the W3C specification. The XML Schema Definition
 * is defined as:
 * <pre><code>
 * &lt;element name="Signature" type="ds:SignatureType"/&gt;
 * &lt;complexType name="SignatureType"&gt;
 *    &lt;sequence&gt;
 *      &lt;element ref="ds:SignedInfo"/&gt;
 *      &lt;element ref="ds:SignatureValue"/&gt;
 *      &lt;element ref="ds:KeyInfo" minOccurs="0"/&gt;
 *      &lt;element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/&gt;
 *    &lt;/sequence&gt;
 *    &lt;attribute name="Id" type="ID" use="optional"/&gt;
 * &lt;/complexType&gt;
 * </code></pre>
 * <p>
 * An <code>XMLSignature</code> instance may be created by invoking one of the
 * {@link XMLSignatureFactory#newXMLSignature newXMLSignature} methods of the
 * {@link XMLSignatureFactory} class.
 *
 * <p>If the contents of the underlying document containing the
 * <code>XMLSignature</code> are subsequently modified, the behavior is
 * undefined.
 *
 * <p>Note that this class is named <code>XMLSignature</code> rather than
 * <code>Signature</code> to avoid naming clashes with the existing
 * {@link Signature java.security.Signature} class.
 *
 * @see XMLSignatureFactory#newXMLSignature(SignedInfo, KeyInfo)
 * @see XMLSignatureFactory#newXMLSignature(SignedInfo, KeyInfo, List, String, String)
 * @author Joyce L. Leung
 * @author Sean Mullan
 * @author Erwin van der Koogh
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public interface XMLSignature extends XMLStructure {

    /**
     * The XML Namespace URI of the W3C Recommendation for XML-Signature
```

```
     * Syntax and Processing.
     */
    final static String XMLNS = "http://www.w3.org/2000/09/xmldsig#";

    /**
     * Validates the signature according to the
     * <a href="http://www.w3.org/TR/xmldsig-core/#sec-CoreValidation">
     * core validation processing rules</a>. This method validates the
     * signature using the existing state, it does not unmarshal and
     * reinitialize the contents of the <code>XMLSignature</code> using the
     * location information specified in the context.
     *
     * <p>This method only validates the signature the first time it is
     * invoked. On subsequent invocations, it returns a cached result.
     *
     * @param validateContext the validating context
     * @return <code>true</code> if the signature passed core validation,
     *    otherwise <code>false</code>
     * @throws ClassCastException if the type of <code>validateContext</code>
     *    is not compatible with this <code>XMLSignature</code>
     * @throws NullPointerException if <code>validateContext</code> is
     *    <code>null</code>
     * @throws XMLSignatureException if an unexpected error occurs during
     *    validation that prevented the validation operation from completing
     */
    boolean validate(XMLValidateContext validateContext)
        throws XMLSignatureException;

    /**
     * Returns the key info of this <code>XMLSignature</code>.
     *
     * @return the key info (may be <code>null</code> if not specified)
     */
    KeyInfo getKeyInfo();

    /**
     * Returns the signed info of this <code>XMLSignature</code>.
     *
     * @return the signed info (never <code>null</code>)
     */
    SignedInfo getSignedInfo();

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of {@link XMLObject}s contained in this <code>XMLSignature</code>.
     *
     * @return an unmodifiable list of <code>XMLObject</code>s (may be empty
     *    but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    List getObjects();

    /**
```

```
     * Returns the optional Id of this <code>XMLSignature</code>.
     *
     * @return the Id (may be <code>null</code> if not specified)
     */
    String getId();

    /**
     * Returns the signature value of this <code>XMLSignature</code>.
     *
     * @return the signature value
     */
    SignatureValue getSignatureValue();

    /**
     * Signs this <code>XMLSignature</code>.
     *
     * <p>If this method throws an exception, this <code>XMLSignature</code> and
     * the <code>signContext</code> parameter will be left in the state that
     * it was in prior to the invocation.
     *
     * @param signContext the signing context
     * @throws ClassCastException if the type of <code>signContext</code> is
     *    not compatible with this <code>XMLSignature</code>
     * @throws NullPointerException if <code>signContext</code> is
     *    <code>null</code>
     * @throws MarshalException if an exception occurs while marshalling
     * @throws XMLSignatureException if an unexpected exception occurs while
     *    generating the signature
     */
    void sign(XMLSignContext signContext) throws MarshalException,
        XMLSignatureException;

    /**
     * Returns the result of the {@link KeySelector}, if specified, after
     * this <code>XMLSignature</code> has been signed or validated.
     *
     * @return the key selector result, or <code>null</code> if a key
     *    selector has not been specified or this <code>XMLSignature</code>
     *    has not been signed or validated
     */
    KeySelectorResult getKeySelectorResult();

    /**
     * A representation of the XML <code>SignatureValue</code> element as
     * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
     * W3C Recommendation for XML-Signature Syntax and Processing</a>.
     * The XML Schema Definition is defined as:
     * <p>
     * <pre>
     *   &lt;element name="SignatureValue" type="ds:SignatureValueType"/&gt;
     *     &lt;complexType name="SignatureValueType"&gt;
     *       &lt;simpleContent&gt;
     *         &lt;extension base="base64Binary"&gt;
```

```
     *           &lt;attribute name="Id" type="ID" use="optional"/&gt;
     *         &lt;/extension&gt;
     *       &lt;/simpleContent&gt;
     *     &lt;/complexType&gt;
     * </pre>
     *
     * @author Sean Mullan
     * @author JSR 105 Expert Group
     */
    public interface SignatureValue extends XMLStructure {
        /**
         * Returns the optional <code>Id</code> attribute of this
         * <code>SignatureValue</code>, which permits this element to be
         * referenced from elsewhere.
         *
         * @return the <code>Id</code> attribute (may be <code>null</code> if
         *    not specified)
         */
        String getId();

        /**
         * Returns the signature value of this <code>SignatureValue</code>.
         *
         * @return the signature value (may be <code>null</code> if the
         *    <code>XMLSignature</code> has not been signed yet). Each
         *    invocation of this method returns a new clone of the array to
         *    prevent subsequent modification.
         */
        byte[] getValue();

        /**
         * Validates the signature value. This method performs a
         * cryptographic validation of the signature calculated over the
         * <code>SignedInfo</code> of the <code>XMLSignature</code>.
         *
         * <p>This method only validates the signature the first
         * time it is invoked. On subsequent invocations, it returns a cached
         * result.
         *
         * @return <code>true</code> if the signature was
         *    validated successfully; <code>false</code> otherwise
         * @param validateContext the validating context
         * @throws NullPointerException if <code>validateContext</code> is
         *    <code>null</code>
         * @throws XMLSignatureException if an unexpected exception occurs while
         *    validating the signature
         */
        boolean validate(XMLValidateContext validateContext)
            throws XMLSignatureException;
    }
}
```

10.15 XMLSignatureException.java

Secuencia 122: javax/xml/crypto/dsig/XMLSignatureException.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XMLSignatureException.java,v 1.5 2005/05/10 16:03:48 mullan Exp $
 */
package javax.xml.crypto.dsig;

import java.io.PrintStream;
import java.io.PrintWriter;

/**
 * Indicates an exceptional condition that occurred during the XML
 * signature generation or validation process.
 *
 * <p>An <code>XMLSignatureException</code> can contain a cause: another
 * throwable that caused this <code>XMLSignatureException</code> to get thrown.
 *
 * @since 1.6
 */
public class XMLSignatureException extends Exception {

    private static final long serialVersionUID = -3438102491013869995L;

    /**
     * The throwable that caused this exception to get thrown, or null if this
     * exception was not caused by another throwable or if the causative
     * throwable is unknown.
     *
```

```
     * @serial
     */
    private Throwable cause;

    /**
     * Constructs a new <code>XMLSignatureException</code> with
     * <code>null</code> as its detail message.
     */
    public XMLSignatureException() {
        super();
    }

    /**
     * Constructs a new <code>XMLSignatureException</code> with the specified
     * detail message.
     *
     * @param message the detail message
     */
    public XMLSignatureException(String message) {
        super(message);
    }

    /**
     * Constructs a new <code>XMLSignatureException</code> with the
     * specified detail message and cause.
     * <p>Note that the detail message associated with
     * <code>cause</code> is <i>not</i> automatically incorporated in
     * this exception's detail message.
     *
     * @param message the detail message
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public XMLSignatureException(String message, Throwable cause) {
        super(message);
        this.cause = cause;
    }

    /**
     * Constructs a new <code>XMLSignatureException</code> with the specified
     * cause and a detail message of
     * <code>(cause==null ? null : cause.toString())</code>
     * (which typically contains the class and detail message of
     * <code>cause</code>).
     *
     * @param cause the cause (A <tt>null</tt> value is permitted, and
     *        indicates that the cause is nonexistent or unknown.)
     */
    public XMLSignatureException(Throwable cause) {
        super(cause==null ? null : cause.toString());
        this.cause = cause;
    }

```

```
    /**
     * Returns the cause of this <code>XMLSignatureException</code> or
     * <code>null</code> if the cause is nonexistent or unknown.  (The
     * cause is the throwable that caused this
     * <code>XMLSignatureException</code> to get thrown.)
     *
     * @return the cause of this <code>XMLSignatureException</code> or
     *         <code>null</code> if the cause is nonexistent or unknown.
     */
    public Throwable getCause() {
        return cause;
    }

    /**
     * Prints this <code>XMLSignatureException</code>, its backtrace and
     * the cause's backtrace to the standard error stream.
     */
    public void printStackTrace() {
        super.printStackTrace();
        if (cause != null) {
            cause.printStackTrace();
        }
    }

    /**
     * Prints this <code>XMLSignatureException</code>, its backtrace and
     * the cause's backtrace to the specified print stream.
     *
     * @param s <code>PrintStream</code> to use for output
     */
    public void printStackTrace(PrintStream s) {
        super.printStackTrace(s);
        if (cause != null) {
            cause.printStackTrace(s);
        }
    }

    /**
     * Prints this <code>XMLSignatureException</code>, its backtrace and
     * the cause's backtrace to the specified print writer.
     *
     * @param s <code>PrintWriter</code> to use for output
     */
    public void printStackTrace(PrintWriter s) {
        super.printStackTrace(s);
        if (cause != null) {
            cause.printStackTrace(s);
        }
    }
}
```

10.16 XMLSignatureFactory.java

Secuencia 123: javax/xml/crypto/dsig/XMLSignatureFactory.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XMLSignatureFactory.java,v 1.14 2005/09/15 14:29:01 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.Data;
import javax.xml.crypto.MarshalException;
import javax.xml.crypto.NoSuchMechanismException;
import javax.xml.crypto.URIDereferencer;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dom.DOMStructure;
import javax.xml.crypto.dsig.keyinfo.KeyInfo;
import javax.xml.crypto.dsig.keyinfo.KeyInfoFactory;
import javax.xml.crypto.dsig.spec.*;
import javax.xml.crypto.dsig.dom.DOMValidateContext;
import javax.xml.crypto.dsig.dom.DOMSignContext;

import java.security.InvalidAlgorithmParameterException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Provider;
import java.security.Security;
import java.util.List;

import sun.security.jca.*;
import sun.security.jca.GetInstance.Instance;

/**
```

```
* A factory for creating {@link XMLSignature} objects from scratch or
* for unmarshalling an <code>XMLSignature</code> object from a corresponding
* XML representation.
*
* <h2>XMLSignatureFactory Type</h2>
*
* <p>Each instance of <code>XMLSignatureFactory</code> supports a specific
* XML mechanism type. To create an <code>XMLSignatureFactory</code>, call one
* of the static {@link #getInstance getInstance} methods, passing in the XML
* mechanism type desired, for example:
*
* <blockquote><code>
* XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
* </code></blockquote>
*
* <p>The objects that this factory produces will be based
* on DOM and abide by the DOM interoperability requirements as defined in the
* <a href="../../../../../technotes/guides/security/xmldsig/overview.html#DOM Mechanism
    Requirements">
* DOM Mechanism Requirements</a> section of the API overview. See the
* <a href="../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
* Service Providers</a> section of the API overview for a list of standard
* mechanism types.
*
* <p><code>XMLSignatureFactory</code> implementations are registered and loaded
* using the {@link java.security.Provider} mechanism.
* For example, a service provider that supports the
* DOM mechanism would be specified in the <code>Provider</code> subclass as:
* <pre>
*     put("XMLSignatureFactory.DOM", "org.example.DOMXMLSignatureFactory");
* </pre>
*
* <p>An implementation MUST minimally support the default mechanism type: DOM.
*
* <p>Note that a caller must use the same <code>XMLSignatureFactory</code>
* instance to create the <code>XMLStructure</code>s of a particular
* <code>XMLSignature</code> that is to be generated. The behavior is
* undefined if <code>XMLStructure</code>s from different providers or
* different mechanism types are used together.
*
* <p>Also, the <code>XMLStructure</code>s that are created by this factory
* may contain state specific to the <code>XMLSignature</code> and are not
* intended to be reusable.
*
* <h2>Creating XMLSignatures from scratch</h2>
*
* <p>Once the <code>XMLSignatureFactory</code> has been created, objects
* can be instantiated by calling the appropriate method. For example, a
* {@link Reference} instance may be created by invoking one of the
* {@link #newReference newReference} methods.
*
* <h2>Unmarshalling XMLSignatures from XML</h2>
*
```

```
 * <p>Alternatively, an <code>XMLSignature</code> may be created from an
 * existing XML representation by invoking the {@link #unmarshalXMLSignature
 * unmarshalXMLSignature} method and passing it a mechanism-specific
 * {@link XMLValidateContext} instance containing the XML content:
 *
 * <pre>
 * DOMValidateContext context = new DOMValidateContext(key, signatureElement);
 * XMLSignature signature = factory.unmarshalXMLSignature(context);
 * </pre>
 *
 * Each <code>XMLSignatureFactory</code> must support the required
 * <code>XMLValidateContext</code> types for that factory type, but may support
 * others. A DOM <code>XMLSignatureFactory</code> must support {@link
 * DOMValidateContext} objects.
 *
 * <h2>Signing and marshalling XMLSignatures to XML</h2>
 *
 * Each <code>XMLSignature</code> created by the factory can also be
 * marshalled to an XML representation and signed, by invoking the
 * {@link XMLSignature#sign sign} method of the
 * {@link XMLSignature} object and passing it a mechanism-specific
 * {@link XMLSignContext} object containing the signing key and
 * marshalling parameters (see {@link DOMSignContext}).
 * For example:
 *
 * <pre>
 *    DOMSignContext context = new DOMSignContext(privateKey, document);
 *    signature.sign(context);
 * </pre>
 *
 * <b>Concurrent Access</b>
 * <p>The static methods of this class are guaranteed to be thread-safe.
 * Multiple threads may concurrently invoke the static methods defined in this
 * class with no ill effects.
 *
 * <p>However, this is not true for the non-static methods defined by this
 * class. Unless otherwise documented by a specific provider, threads that
 * need to access a single <code>XMLSignatureFactory</code> instance
 * concurrently should synchronize amongst themselves and provide the
 * necessary locking. Multiple threads each manipulating a different
 * <code>XMLSignatureFactory</code> instance need not synchronize.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public abstract class XMLSignatureFactory {

    private String mechanismType;
    private Provider provider;

    /**
     * Default constructor, for invocation by subclasses.
```

```
     */
    protected XMLSignatureFactory() {}

    /**
     * Returns an <code>XMLSignatureFactory</code> that supports the
     * specified XML processing mechanism and representation type (ex: "DOM").
     *
     * <p>This method uses the standard JCA provider lookup mechanism to
     * locate and instantiate an <code>XMLSignatureFactory</code>
     * implementation of the desired mechanism type. It traverses the list of
     * registered security <code>Provider</code>s, starting with the most
     * preferred <code>Provider</code>.  A new <code>XMLSignatureFactory</code>
     * object from the first <code>Provider</code> that supports the specified
     * mechanism is returned.
     *
     * <p>Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
     *    Service Providers</a> section of the API overview for a list of
     *    standard mechanism types.
     * @return a new <code>XMLSignatureFactory</code>
     * @throws NullPointerException if <code>mechanismType</code> is
     *    <code>null</code>
     * @throws NoSuchMechanismException if no <code>Provider</code> supports an
     *    <code>XMLSignatureFactory</code> implementation for the specified
     *    mechanism
     * @see Provider
     */
    public static XMLSignatureFactory getInstance(String mechanismType) {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        }
        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("XMLSignatureFactory", null, mechanismType);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        XMLSignatureFactory factory = (XMLSignatureFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns an <code>XMLSignatureFactory</code> that supports the
     * requested XML processing mechanism and representation type (ex: "DOM"),
     * as supplied by the specified provider. Note that the specified
     * <code>Provider</code> object does not have to be registered in the
```

```
     * provider list.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
     *    Service Providers</a> section of the API overview for a list of
     *    standard mechanism types.
     * @param provider the <code>Provider</code> object
     * @return a new <code>XMLSignatureFactory</code>
     * @throws NullPointerException if <code>provider</code> or
     *    <code>mechanismType</code> is <code>null</code>
     * @throws NoSuchMechanismException if an <code>XMLSignatureFactory</code>
     *   implementation for the specified mechanism is not available
     *   from the specified <code>Provider</code> object
     * @see Provider
     */
    public static XMLSignatureFactory getInstance(String mechanismType,
        Provider provider) {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        } else if (provider == null) {
            throw new NullPointerException("provider cannot be null");
        }

        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("XMLSignatureFactory", null, mechanismType, provider);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        XMLSignatureFactory factory = (XMLSignatureFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns an <code>XMLSignatureFactory</code> that supports the
     * requested XML processing mechanism and representation type (ex: "DOM"),
     * as supplied by the specified provider. The specified provider must be
     * registered in the security provider list.
     *
     * <p>Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
     *    Service Providers</a> section of the API overview for a list of
     *    standard mechanism types.
     * @param provider the string name of the provider
     * @return a new <code>XMLSignatureFactory</code>
```

```
     * @throws NoSuchProviderException if the specified provider is not
     *    registered in the security provider list
     * @throws NullPointerException if <code>provider</code> or
     *    <code>mechanismType</code> is <code>null</code>
     * @throws NoSuchMechanismException if an <code>XMLSignatureFactory</code>
     *    implementation for the specified mechanism is not
     *    available from the specified provider
     * @see Provider
     */
    public static XMLSignatureFactory getInstance(String mechanismType,
        String provider) throws NoSuchProviderException {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        } else if (provider == null) {
            throw new NullPointerException("provider cannot be null");
        } else if (provider.length() == 0) {
            throw new NoSuchProviderException();
        }

        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("XMLSignatureFactory", null, mechanismType, provider);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        XMLSignatureFactory factory = (XMLSignatureFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns an <code>XMLSignatureFactory</code> that supports the
     * default XML processing mechanism and representation type ("DOM").
     *
     * <p>This method uses the standard JCA provider lookup mechanism to
     * locate and instantiate an <code>XMLSignatureFactory</code>
     * implementation of the default mechanism type. It traverses the list of
     * registered security <code>Provider</code>s, starting with the most
     * preferred <code>Provider</code>.  A new <code>XMLSignatureFactory</code>
     * object from the first <code>Provider</code> that supports the DOM
     * mechanism is returned.
     *
     * <p>Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @return a new <code>XMLSignatureFactory</code>
     * @throws NoSuchMechanismException if no <code>Provider</code> supports an
     *    <code>XMLSignatureFactory</code> implementation for the DOM
     *    mechanism
     * @see Provider
     */
```

```
    public static XMLSignatureFactory getInstance() {
        return getInstance("DOM");
    }

    /**
     * Returns the type of the XML processing mechanism and representation
     * supported by this <code>XMLSignatureFactory</code> (ex: "DOM").
     *
     * @return the XML processing mechanism type supported by this
     *    <code>XMLSignatureFactory</code>
     */
    public final String getMechanismType() {
        return mechanismType;
    }

    /**
     * Returns the provider of this <code>XMLSignatureFactory</code>.
     *
     * @return the provider of this <code>XMLSignatureFactory</code>
     */
    public final Provider getProvider() {
        return provider;
    }

    /**
     * Creates an <code>XMLSignature</code> and initializes it with the contents
     * of the specified <code>SignedInfo</code> and <code>KeyInfo</code>
     * objects.
     *
     * @param si the signed info
     * @param ki the key info (may be <code>null</code>)
     * @return an <code>XMLSignature</code>
     * @throws NullPointerException if <code>si</code> is <code>null</code>
     */
    public abstract XMLSignature newXMLSignature(SignedInfo si, KeyInfo ki);

    /**
     * Creates an <code>XMLSignature</code> and initializes it with the
     * specified parameters.
     *
     * @param si the signed info
     * @param ki the key info (may be <code>null</code>)
     * @param objects a list of {@link XMLObject}s (may be empty or
     *    <code>null</code>)
     * @param id the Id (may be <code>null</code>)
     * @param signatureValueId the SignatureValue Id (may be <code>null</code>)
     * @return an <code>XMLSignature</code>
     * @throws NullPointerException if <code>si</code> is <code>null</code>
     * @throws ClassCastException if any of the <code>objects</code> are not of
     *    type <code>XMLObject</code>
     */
    @SuppressWarnings("rawtypes")
    public abstract XMLSignature newXMLSignature(SignedInfo si, KeyInfo ki,
```

```
        List objects, String id, String signatureValueId);

    /**
     * Creates a <code>Reference</code> with the specified URI and digest
     * method.
     *
     * @param uri the reference URI (may be <code>null</code>)
     * @param dm the digest method
     * @return a <code>Reference</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant
     * @throws NullPointerException if <code>dm</code> is <code>null</code>
     */
    public abstract Reference newReference(String uri, DigestMethod dm);

    /**
     * Creates a <code>Reference</code> with the specified parameters.
     *
     * @param uri the reference URI (may be <code>null</code>)
     * @param dm the digest method
     * @param transforms a list of {@link Transform}s. The list is defensively
     *    copied to protect against subsequent modification. May be
     *    <code>null</code> or empty.
     * @param type the reference type, as a URI (may be <code>null</code>)
     * @param id the reference ID (may be <code>null</code>)
     * @return a <code>Reference</code>
     * @throws ClassCastException if any of the <code>transforms</code> are
     *    not of type <code>Transform</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant
     * @throws NullPointerException if <code>dm</code> is <code>null</code>
     */
    @SuppressWarnings("rawtypes")
    public abstract Reference newReference(String uri, DigestMethod dm,
        List transforms, String type, String id);

    /**
     * Creates a <code>Reference</code> with the specified parameters and
     * pre-calculated digest value.
     *
     * <p>This method is useful when the digest value of a
     * <code>Reference</code> has been previously computed. See for example,
     * the
     * <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss">
     * OASIS-DSS (Digital Signature Services)</a> specification.
     *
     * @param uri the reference URI (may be <code>null</code>)
     * @param dm the digest method
     * @param transforms a list of {@link Transform}s. The list is defensively
     *    copied to protect against subsequent modification. May be
     *    <code>null</code> or empty.
     * @param type the reference type, as a URI (may be <code>null</code>)
     * @param id the reference ID (may be <code>null</code>)
```

```
     * @param digestValue the digest value. The array is cloned to protect
     *    against subsequent modification.
     * @return a <code>Reference</code>
     * @throws ClassCastException if any of the <code>transforms</code> are
     *    not of type <code>Transform</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant
     * @throws NullPointerException if <code>dm</code> or
     *    <code>digestValue</code> is <code>null</code>
     */
    @SuppressWarnings("rawtypes")
    public abstract Reference newReference(String uri, DigestMethod dm,
        List transforms, String type, String id, byte[] digestValue);

    /**
     * Creates a <code>Reference</code> with the specified parameters.
     *
     * <p>This method is useful when a list of transforms have already been
     * applied to the <code>Reference</code>. See for example,
     * the
     * <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss">
     * OASIS-DSS (Digital Signature Services)</a> specification.
     *
     * <p>When an <code>XMLSignature</code> containing this reference is
     * generated, the specified <code>transforms</code> (if non-null) are
     * applied to the specified <code>result</code>. The
     * <code>Transforms</code> element of the resulting <code>Reference</code>
     * element is set to the concatenation of the
     * <code>appliedTransforms</code> and <code>transforms</code>.
     *
     * @param uri the reference URI (may be <code>null</code>)
     * @param dm the digest method
     * @param appliedTransforms a list of {@link Transform}s that have
     *    already been applied. The list is defensively
     *    copied to protect against subsequent modification. The list must
     *    contain at least one entry.
     * @param result the result of processing the sequence of
     *    <code>appliedTransforms</code>
     * @param transforms a list of {@link Transform}s that are to be applied
     *    when generating the signature. The list is defensively copied to
     *    protect against subsequent modification. May be <code>null</code>
     *    or empty.
     * @param type the reference type, as a URI (may be <code>null</code>)
     * @param id the reference ID (may be <code>null</code>)
     * @return a <code>Reference</code>
     * @throws ClassCastException if any of the transforms (in either list)
     *    are not of type <code>Transform</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant or <code>appliedTransforms</code> is empty
     * @throws NullPointerException if <code>dm</code>,
     *    <code>appliedTransforms</code> or <code>result</code> is
     *    <code>null</code>
     */
```

```
    @SuppressWarnings("rawtypes")
    public abstract Reference newReference(String uri, DigestMethod dm,
        List appliedTransforms, Data result, List transforms, String type,
        String id);

    /**
     * Creates a <code>SignedInfo</code> with the specified canonicalization
     * and signature methods, and list of one or more references.
     *
     * @param cm the canonicalization method
     * @param sm the signature method
     * @param references a list of one or more {@link Reference}s. The list is
     *    defensively copied to protect against subsequent modification.
     * @return a <code>SignedInfo</code>
     * @throws ClassCastException if any of the references are not of
     *    type <code>Reference</code>
     * @throws IllegalArgumentException if <code>references</code> is empty
     * @throws NullPointerException if any of the parameters
     *    are <code>null</code>
     */
    @SuppressWarnings("rawtypes")
    public abstract SignedInfo newSignedInfo(CanonicalizationMethod cm,
        SignatureMethod sm, List references);

    /**
     * Creates a <code>SignedInfo</code> with the specified parameters.
     *
     * @param cm the canonicalization method
     * @param sm the signature method
     * @param references a list of one or more {@link Reference}s. The list is
     *    defensively copied to protect against subsequent modification.
     * @param id the id (may be <code>null</code>)
     * @return a <code>SignedInfo</code>
     * @throws ClassCastException if any of the references are not of
     *    type <code>Reference</code>
     * @throws IllegalArgumentException if <code>references</code> is empty
     * @throws NullPointerException if <code>cm</code>, <code>sm</code>, or
     *    <code>references</code> are <code>null</code>
     */
    @SuppressWarnings("rawtypes")
    public abstract SignedInfo newSignedInfo(CanonicalizationMethod cm,
        SignatureMethod sm, List references, String id);

    // Object factory methods
    /**
     * Creates an <code>XMLObject</code> from the specified parameters.
     *
     * @param content a list of {@link XMLStructure}s. The list
     *    is defensively copied to protect against subsequent modification.
     *    May be <code>null</code> or empty.
     * @param id the Id (may be <code>null</code>)
     * @param mimeType the mime type (may be <code>null</code>)
     * @param encoding the encoding (may be <code>null</code>)
```

```
     * @return an <code>XMLObject</code>
     * @throws ClassCastException if <code>content</code> contains any
     *    entries that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract XMLObject newXMLObject(List content, String id,
        String mimeType, String encoding);

    /**
     * Creates a <code>Manifest</code> containing the specified
     * list of {@link Reference}s.
     *
     * @param references a list of one or more <code>Reference</code>s. The list
     *    is defensively copied to protect against subsequent modification.
     * @return a <code>Manifest</code>
     * @throws NullPointerException if <code>references</code> is
     *    <code>null</code>
     * @throws IllegalArgumentException if <code>references</code> is empty
     * @throws ClassCastException if <code>references</code> contains any
     *    entries that are not of type {@link Reference}
     */
    @SuppressWarnings("rawtypes")
    public abstract Manifest newManifest(List references);

    /**
     * Creates a <code>Manifest</code> containing the specified
     * list of {@link Reference}s and optional id.
     *
     * @param references a list of one or more <code>Reference</code>s. The list
     *    is defensively copied to protect against subsequent modification.
     * @param id the id (may be <code>null</code>)
     * @return a <code>Manifest</code>
     * @throws NullPointerException if <code>references</code> is
     *    <code>null</code>
     * @throws IllegalArgumentException if <code>references</code> is empty
     * @throws ClassCastException if <code>references</code> contains any
     *    entries that are not of type {@link Reference}
     */
    @SuppressWarnings("rawtypes")
    public abstract Manifest newManifest(List references, String id);

    /**
     * Creates a <code>SignatureProperty</code> containing the specified
     * list of {@link XMLStructure}s, target URI and optional id.
     *
     * @param content a list of one or more <code>XMLStructure</code>s. The list
     *    is defensively copied to protect against subsequent modification.
     * @param target the target URI of the Signature that this property applies
     *    to
     * @param id the id (may be <code>null</code>)
     * @return a <code>SignatureProperty</code>
     * @throws NullPointerException if <code>content</code> or
     *    <code>target</code> is <code>null</code>
```

```
     * @throws IllegalArgumentException if <code>content</code> is empty
     * @throws ClassCastException if <code>content</code> contains any
     *    entries that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract SignatureProperty newSignatureProperty
        (List content, String target, String id);

    /**
     * Creates a <code>SignatureProperties</code> containing the specified
     * list of {@link SignatureProperty}s and optional id.
     *
     * @param properties a list of one or more <code>SignatureProperty</code>s.
     *    The list is defensively copied to protect against subsequent
     *    modification.
     * @param id the id (may be <code>null</code>)
     * @return a <code>SignatureProperties</code>
     * @throws NullPointerException if <code>properties</code>
     *    is <code>null</code>
     * @throws IllegalArgumentException if <code>properties</code> is empty
     * @throws ClassCastException if <code>properties</code> contains any
     *    entries that are not of type {@link SignatureProperty}
     */
    @SuppressWarnings("rawtypes")
    public abstract SignatureProperties newSignatureProperties
        (List properties, String id);

    // Algorithm factory methods
    /**
     * Creates a <code>DigestMethod</code> for the specified algorithm URI
     * and parameters.
     *
     * @param algorithm the URI identifying the digest algorithm
     * @param params algorithm-specific digest parameters (may be
     *    <code>null</code>)
     * @return the <code>DigestMethod</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
    public abstract DigestMethod newDigestMethod(String algorithm,
        DigestMethodParameterSpec params) throws NoSuchAlgorithmException,
        InvalidAlgorithmParameterException;

    /**
     * Creates a <code>SignatureMethod</code> for the specified algorithm URI
     * and parameters.
     *
     * @param algorithm the URI identifying the signature algorithm
     * @param params algorithm-specific signature parameters (may be
```

```
     *    <code>null</code>)
     * @return the <code>SignatureMethod</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
    public abstract SignatureMethod newSignatureMethod(String algorithm,
        SignatureMethodParameterSpec params) throws NoSuchAlgorithmException,
        InvalidAlgorithmParameterException;

    /**
     * Creates a <code>Transform</code> for the specified algorithm URI
     * and parameters.
     *
     * @param algorithm the URI identifying the transform algorithm
     * @param params algorithm-specific transform parameters (may be
     *    <code>null</code>)
     * @return the <code>Transform</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
    public abstract Transform newTransform(String algorithm,
        TransformParameterSpec params) throws NoSuchAlgorithmException,
        InvalidAlgorithmParameterException;

    /**
     * Creates a <code>Transform</code> for the specified algorithm URI
     * and parameters. The parameters are specified as a mechanism-specific
     * <code>XMLStructure</code> (ex: {@link DOMStructure}). This method is
     * useful when the parameters are in XML form or there is no standard
     * class for specifying the parameters.
     *
     * @param algorithm the URI identifying the transform algorithm
     * @param params a mechanism-specific XML structure from which to
     *   unmarshal the parameters from (may be <code>null</code> if
     *   not required or optional)
     * @return the <code>Transform</code>
     * @throws ClassCastException if the type of <code>params</code> is
     *   inappropriate for this <code>XMLSignatureFactory</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
```

```java
    public abstract Transform newTransform(String algorithm,
        XMLStructure params) throws NoSuchAlgorithmException,
        InvalidAlgorithmParameterException;

    /**
     * Creates a <code>CanonicalizationMethod</code> for the specified
     * algorithm URI and parameters.
     *
     * @param algorithm the URI identifying the canonicalization algorithm
     * @param params algorithm-specific canonicalization parameters (may be
     *    <code>null</code>)
     * @return the <code>CanonicalizationMethod</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
    public abstract CanonicalizationMethod newCanonicalizationMethod(
        String algorithm, C14NMethodParameterSpec params)
        throws NoSuchAlgorithmException, InvalidAlgorithmParameterException;

    /**
     * Creates a <code>CanonicalizationMethod</code> for the specified
     * algorithm URI and parameters. The parameters are specified as a
     * mechanism-specific <code>XMLStructure</code> (ex: {@link DOMStructure}).
     * This method is useful when the parameters are in XML form or there is
     * no standard class for specifying the parameters.
     *
     * @param algorithm the URI identifying the canonicalization algorithm
     * @param params a mechanism-specific XML structure from which to
     *   unmarshal the parameters from (may be <code>null</code> if
     *   not required or optional)
     * @return the <code>CanonicalizationMethod</code>
     * @throws ClassCastException if the type of <code>params</code> is
     *   inappropriate for this <code>XMLSignatureFactory</code>
     * @throws InvalidAlgorithmParameterException if the specified parameters
     *    are inappropriate for the requested algorithm
     * @throws NoSuchAlgorithmException if an implementation of the
     *    specified algorithm cannot be found
     * @throws NullPointerException if <code>algorithm</code> is
     *    <code>null</code>
     */
    public abstract CanonicalizationMethod newCanonicalizationMethod(
        String algorithm, XMLStructure params)
        throws NoSuchAlgorithmException, InvalidAlgorithmParameterException;

    /**
     * Returns a <code>KeyInfoFactory</code> that creates <code>KeyInfo</code>
     * objects. The returned <code>KeyInfoFactory</code> has the same
     * mechanism type and provider as this <code>XMLSignatureFactory</code>.
     *
```

```
     * @return a <code>KeyInfoFactory</code>
     * @throws NoSuchMechanismException if a <code>KeyFactory</code>
     *    implementation with the same mechanism type and provider
     *    is not available
     */
    public final KeyInfoFactory getKeyInfoFactory() {
        return KeyInfoFactory.getInstance(getMechanismType(), getProvider());
    }

    /**
     * Unmarshals a new <code>XMLSignature</code> instance from a
     * mechanism-specific <code>XMLValidateContext</code> instance.
     *
     * @param context a mechanism-specific context from which to unmarshal the
     *    signature from
     * @return the <code>XMLSignature</code>
     * @throws NullPointerException if <code>context</code> is
     *    <code>null</code>
     * @throws ClassCastException if the type of <code>context</code> is
     *    inappropriate for this factory
     * @throws MarshalException if an unrecoverable exception occurs
     *    during unmarshalling
     */
    public abstract XMLSignature unmarshalXMLSignature
        (XMLValidateContext context) throws MarshalException;

    /**
     * Unmarshals a new <code>XMLSignature</code> instance from a
     * mechanism-specific <code>XMLStructure</code> instance.
     * This method is useful if you only want to unmarshal (and not
     * validate) an <code>XMLSignature</code>.
     *
     * @param xmlStructure a mechanism-specific XML structure from which to
     *    unmarshal the signature from
     * @return the <code>XMLSignature</code>
     * @throws NullPointerException if <code>xmlStructure</code> is
     *    <code>null</code>
     * @throws ClassCastException if the type of <code>xmlStructure</code> is
     *    inappropriate for this factory
     * @throws MarshalException if an unrecoverable exception occurs
     *    during unmarshalling
     */
    public abstract XMLSignature unmarshalXMLSignature
        (XMLStructure xmlStructure) throws MarshalException;

    /**
     * Indicates whether a specified feature is supported.
     *
     * @param feature the feature name (as an absolute URI)
     * @return <code>true</code> if the specified feature is supported,
     *    <code>false</code> otherwise
     * @throws NullPointerException if <code>feature</code> is <code>null</code>
     */
```

```
    public abstract boolean isFeatureSupported(String feature);

    /**
     * Returns a reference to the <code>URIDereferencer</code> that is used by
     * default to dereference URIs in {@link Reference} objects.
     *
     * @return a reference to the default <code>URIDereferencer</code> (never
     *    <code>null</code>)
     */
    public abstract URIDereferencer getURIDereferencer();
}
```

10.17 XMLValidateContext.java

Secuencia 124: javax/xml/crypto/dsig/XMLValidateContext.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XMLValidateContext.java,v 1.8 2005/05/10 16:03:49 mullan Exp $
 */
package javax.xml.crypto.dsig;

import javax.xml.crypto.XMLCryptoContext;

/**
 * Contains context information for validating XML Signatures. This interface
 * is primarily intended for type-safety.
 *
 * <p>Note that <code>XMLValidateContext</code> instances can contain
 * information and state specific to the XML signature structure it is
 * used with. The results are unpredictable if an
```

```
 * <code>XMLValidateContext</code> is used with different signature structures
 * (for example, you should not use the same <code>XMLValidateContext</code>
 * instance to validate two different {@link XMLSignature} objects).
 * <p>
 * <b><a name="Supported Properties"></a>Supported Properties</b>
 * <p>The following properties can be set by an application using the
 * {@link #setProperty setProperty} method.
 * <ul>
 *   <li><code>javax.xml.crypto.dsig.cacheReference</code>: value must be a
 *      {@link Boolean}. This property controls whether or not the
 *      {@link Reference#validate Reference.validate} method will cache the
 *      dereferenced content and pre-digested input for subsequent retrieval via
 *      the {@link Reference#getDereferencedData Reference.getDereferencedData}
 *      and {@link Reference#getDigestInputStream
 *      Reference.getDigestInputStream} methods. The default value if not
 *      specified is <code>Boolean.FALSE</code>.
 * </ul>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignature#validate(XMLValidateContext)
 * @see Reference#validate(XMLValidateContext)
 */
public interface XMLValidateContext extends XMLCryptoContext {}
```

11 Xml Crypto Dsig Dom (javax.xml.crypto.dsig.dom package)

11.1 DOMSignContext.java

Secuencia 125: javax/xml/crypto/dsig/dom/DOMSignContext.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 */
/*
 * $Id: DOMSignContext.java,v 1.9 2005/05/10 16:31:14 mullan Exp $
 */
package javax.xml.crypto.dsig.dom;

import javax.xml.crypto.KeySelector;
import javax.xml.crypto.dom.DOMCryptoContext;
import javax.xml.crypto.dsig.XMLSignContext;
import javax.xml.crypto.dsig.XMLSignature;
import java.security.Key;
import org.w3c.dom.Node;

/**
 * A DOM-specific {@link XMLSignContext}. This class contains additional methods
 * to specify the location in a DOM tree where an {@link XMLSignature}
 * object is to be marshalled when generating the signature.
 *
 * <p>Note that <code>DOMSignContext</code> instances can contain
 * information and state specific to the XML signature structure it is
 * used with. The results are unpredictable if a
 * <code>DOMSignContext</code> is used with different signature structures
 * (for example, you should not use the same <code>DOMSignContext</code>
 * instance to sign two different {@link XMLSignature} objects).
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public class DOMSignContext extends DOMCryptoContext implements XMLSignContext {

    private Node parent;
    private Node nextSibling;

    /**
     * Creates a <code>DOMSignContext</code> with the specified signing key
     * and parent node. The signing key is stored in a
     * {@link KeySelector#singletonKeySelector singleton KeySelector} that is
     * returned by the {@link #getKeySelector getKeySelector} method.
     * The marshalled <code>XMLSignature</code> will be added as the last
     * child element of the specified parent node unless a next sibling node is
     * specified by invoking the {@link #setNextSibling setNextSibling} method.
     *
     * @param signingKey the signing key
     * @param parent the parent node
     * @throws NullPointerException if <code>signingKey</code> or
     *    <code>parent</code> is <code>null</code>
     */
    public DOMSignContext(Key signingKey, Node parent) {
        if (signingKey == null) {
            throw new NullPointerException("signingKey cannot be null");
        }
```

```
        if (parent == null) {
            throw new NullPointerException("parent cannot be null");
        }
        setKeySelector(KeySelector.singletonKeySelector(signingKey));
        this.parent = parent;
    }

    /**
     * Creates a <code>DOMSignContext</code> with the specified signing key,
     * parent and next sibling nodes. The signing key is stored in a
     * {@link KeySelector#singletonKeySelector singleton KeySelector} that is
     * returned by the {@link #getKeySelector getKeySelector} method.
     * The marshalled <code>XMLSignature</code> will be inserted as a child
     * element of the specified parent node and immediately before the
     * specified next sibling node.
     *
     * @param signingKey the signing key
     * @param parent the parent node
     * @param nextSibling the next sibling node
     * @throws NullPointerException if <code>signingKey</code>,
     *    <code>parent</code> or <code>nextSibling</code> is <code>null</code>
     */
    public DOMSignContext(Key signingKey, Node parent, Node nextSibling) {
        if (signingKey == null) {
            throw new NullPointerException("signingKey cannot be null");
        }
        if (parent == null) {
            throw new NullPointerException("parent cannot be null");
        }
        if (nextSibling == null) {
            throw new NullPointerException("nextSibling cannot be null");
        }
        setKeySelector(KeySelector.singletonKeySelector(signingKey));
        this.parent = parent;
        this.nextSibling = nextSibling;
    }

    /**
     * Creates a <code>DOMSignContext</code> with the specified key selector
     * and parent node. The marshalled <code>XMLSignature</code> will be added
     * as the last child element of the specified parent node unless a next
     * sibling node is specified by invoking the
     * {@link #setNextSibling setNextSibling} method.
     *
     * @param ks the key selector
     * @param parent the parent node
     * @throws NullPointerException if <code>ks</code> or <code>parent</code>
     *    is <code>null</code>
     */
    public DOMSignContext(KeySelector ks, Node parent) {
        if (ks == null) {
            throw new NullPointerException("key selector cannot be null");
        }
```

```
        if (parent == null) {
            throw new NullPointerException("parent cannot be null");
        }
        setKeySelector(ks);
        this.parent = parent;
    }

    /**
     * Creates a <code>DOMSignContext</code> with the specified key selector,
     * parent and next sibling nodes. The marshalled <code>XMLSignature</code>
     * will be inserted as a child element of the specified parent node and
     * immediately before the specified next sibling node.
     *
     * @param ks the key selector
     * @param parent the parent node
     * @param nextSibling the next sibling node
     * @throws NullPointerException if <code>ks</code>, <code>parent</code> or
     *    <code>nextSibling</code> is <code>null</code>
     */
    public DOMSignContext(KeySelector ks, Node parent, Node nextSibling) {
        if (ks == null) {
            throw new NullPointerException("key selector cannot be null");
        }
        if (parent == null) {
            throw new NullPointerException("parent cannot be null");
        }
        if (nextSibling == null) {
            throw new NullPointerException("nextSibling cannot be null");
        }
        setKeySelector(ks);
        this.parent = parent;
        this.nextSibling = nextSibling;
    }

    /**
     * Sets the parent node.
     *
     * @param parent the parent node. The marshalled <code>XMLSignature</code>
     *    will be added as a child element of this node.
     * @throws NullPointerException if <code>parent</code> is <code>null</code>
     * @see #getParent
     */
    public void setParent(Node parent) {
        if (parent == null) {
            throw new NullPointerException("parent is null");
        }
        this.parent = parent;
    }

    /**
     * Sets the next sibling node.
     *
     * @param nextSibling the next sibling node. The marshalled
```

```
     *    <code>XMLSignature</code> will be inserted immediately before this
     *    node. Specify <code>null</code> to remove the current setting.
     * @see #getNextSibling
     */
    public void setNextSibling(Node nextSibling) {
        this.nextSibling = nextSibling;
    }

    /**
     * Returns the parent node.
     *
     * @return the parent node (never <code>null</code>)
     * @see #setParent(Node)
     */
    public Node getParent() {
        return parent;
    }

    /**
     * Returns the nextSibling node.
     *
     * @return the nextSibling node, or <code>null</code> if not specified.
     * @see #setNextSibling(Node)
     */
    public Node getNextSibling() {
        return nextSibling;
    }
}
```

11.2 DOMValidateContext.java

Secuencia 126: javax/xml/crypto/dsig/dom/DOMValidateContext.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */
/*
 * $Id: DOMValidateContext.java,v 1.8 2005/05/10 16:31:14 mullan Exp $
 */
package javax.xml.crypto.dsig.dom;

import javax.xml.crypto.KeySelector;
import javax.xml.crypto.dom.DOMCryptoContext;
import javax.xml.crypto.dsig.XMLSignature;
import javax.xml.crypto.dsig.XMLSignatureFactory;
import javax.xml.crypto.dsig.XMLValidateContext;
import java.security.Key;
import org.w3c.dom.Node;

/**
 * A DOM-specific {@link XMLValidateContext}. This class contains additional
 * methods to specify the location in a DOM tree where an {@link XMLSignature}
 * is to be unmarshalled and validated from.
 *
 * <p>Note that the behavior of an unmarshalled <code>XMLSignature</code>
 * is undefined if the contents of the underlying DOM tree are modified by the
 * caller after the <code>XMLSignature</code> is created.
 *
 * <p>Also, note that <code>DOMValidateContext</code> instances can contain
 * information and state specific to the XML signature structure it is
 * used with. The results are unpredictable if a
 * <code>DOMValidateContext</code> is used with different signature structures
 * (for example, you should not use the same <code>DOMValidateContext</code>
 * instance to validate two different {@link XMLSignature} objects).
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XMLSignatureFactory#unmarshalXMLSignature(XMLValidateContext)
 */
public class DOMValidateContext extends DOMCryptoContext
    implements XMLValidateContext {

    private Node node;

    /**
     * Creates a <code>DOMValidateContext</code> containing the specified key
     * selector and node.
     *
     * @param ks a key selector for finding a validation key
     * @param node the node
     * @throws NullPointerException if <code>ks</code> or <code>node</code> is
     *    <code>null</code>
     */
    public DOMValidateContext(KeySelector ks, Node node) {
        if (ks == null) {
```

```
            throw new NullPointerException("key selector is null");
        }
        init(node, ks);
    }

    /**
     * Creates a <code>DOMValidateContext</code> containing the specified key
     * and node. The validating key will be stored in a
     * {@link KeySelector#singletonKeySelector singleton KeySelector} that
     * is returned when the {@link #getKeySelector getKeySelector}
     * method is called.
     *
     * @param validatingKey the validating key
     * @param node the node
     * @throws NullPointerException if <code>validatingKey</code> or
     *    <code>node</code> is <code>null</code>
     */
    public DOMValidateContext(Key validatingKey, Node node) {
        if (validatingKey == null) {
            throw new NullPointerException("validatingKey is null");
        }
        init(node, KeySelector.singletonKeySelector(validatingKey));
    }

    private void init(Node node, KeySelector ks) {
        if (node == null) {
            throw new NullPointerException("node is null");
        }

        this.node = node;
        super.setKeySelector(ks);
        if (System.getSecurityManager() != null) {
            super.setProperty("org.jcp.xml.dsig.secureValidation",
                              Boolean.TRUE);
        }
    }

    /**
     * Sets the node.
     *
     * @param node the node
     * @throws NullPointerException if <code>node</code> is <code>null</code>
     * @see #getNode
     */
    public void setNode(Node node) {
        if (node == null) {
            throw new NullPointerException();
        }
        this.node = node;
    }

    /**
     * Returns the node.
```

```
     *
     * @return the node (never <code>null</code>)
     * @see #setNode(Node)
     */
    public Node getNode() {
        return node;
    }
}
```

12 Xml Crypto Dsig Keyinfo (javax.xml.crypto.dsig.keyinfo package)

12.1 KeyInfo.java

Secuencia 127: javax/xml/crypto/dsig/keyinfo/KeyInfo.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeyInfo.java,v 1.7 2005/05/10 16:35:34 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import java.util.List;
import javax.xml.crypto.MarshalException;
import javax.xml.crypto.XMLCryptoContext;
import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>KeyInfo</code> element as defined in
 * the <a href="http://www.w3.org/TR/xmldsig-core/">
```

```
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * A <code>KeyInfo</code> contains a list of {@link XMLStructure}s, each of
 * which contain information that enables the recipient(s) to obtain the key
 * needed to validate an XML signature. The XML Schema Definition is defined as:
 *
 * <pre>
 * &lt;element name="KeyInfo" type="ds:KeyInfoType"/&gt;
 * &lt;complexType name="KeyInfoType" mixed="true"&gt;
 *   &lt;choice maxOccurs="unbounded"&gt;
 *     &lt;element ref="ds:KeyName"/&gt;
 *     &lt;element ref="ds:KeyValue"/&gt;
 *     &lt;element ref="ds:RetrievalMethod"/&gt;
 *     &lt;element ref="ds:X509Data"/&gt;
 *     &lt;element ref="ds:PGPData"/&gt;
 *     &lt;element ref="ds:SPKIData"/&gt;
 *     &lt;element ref="ds:MgmtData"/&gt;
 *     &lt;any processContents="lax" namespace="##other"/&gt;
 *     &lt;!-- (1,1) elements from (0,unbounded) namespaces --&gt;
 *   &lt;/choice&gt;
 *   &lt;attribute name="Id" type="ID" use="optional"/&gt;
 * &lt;/complexType&gt;
 * </pre>
 *
 * A <code>KeyInfo</code> instance may be created by invoking one of the
 * {@link KeyInfoFactory#newKeyInfo newKeyInfo} methods of the
 * {@link KeyInfoFactory} class, and passing it a list of one or more
 * <code>XMLStructure</code>s and an optional id parameter;
 * for example:
 * <pre>
 *   KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 *   KeyInfo keyInfo = factory.newKeyInfo
 *       (Collections.singletonList(factory.newKeyName("Alice"), "keyinfo-1"));
 * </pre>
 *
 * <p><code>KeyInfo</code> objects can also be marshalled to XML by invoking
 * the {@link #marshal marshal} method.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newKeyInfo(List)
 * @see KeyInfoFactory#newKeyInfo(List, String)
 */
public interface KeyInfo extends XMLStructure {

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} containing the key information. Each entry of the list is
     * an {@link XMLStructure}.
     *
     * <p>If there is a public subclass representing the type of
     * <code>XMLStructure</code>, it is returned as an instance of that
     * class (ex: an <code>X509Data</code> element would be returned as an
```

```
     * instance of {@link javax.xml.crypto.dsig.keyinfo.X509Data}).
     *
     * @return an unmodifiable list of one or more <code>XMLStructure</code>s
     *    in this <code>KeyInfo</code>. Never returns <code>null</code> or an
     *    empty list.
     */
    @SuppressWarnings("rawtypes")
    List getContent();

    /**
     * Return the optional Id attribute of this <code>KeyInfo</code>, which
     * may be useful for referencing this <code>KeyInfo</code> from other
     * XML structures.
     *
     * @return the Id attribute of this <code>KeyInfo</code> (may be
     *    <code>null</code> if not specified)
     */
    String getId();

    /**
     * Marshals the key info to XML.
     *
     * @param parent a mechanism-specific structure containing the parent node
     *    that the marshalled key info will be appended to
     * @param context the <code>XMLCryptoContext</code> containing additional
     *    context (may be null if not applicable)
     * @throws ClassCastException if the type of <code>parent</code> or
     *    <code>context</code> is not compatible with this key info
     * @throws MarshalException if the key info cannot be marshalled
     * @throws NullPointerException if <code>parent</code> is <code>null</code>
     */
    void marshal(XMLStructure parent, XMLCryptoContext context)
        throws MarshalException;
}
```

12.2 KeyInfoFactory.java

Secuencia 128: javax/xml/crypto/dsig/keyinfo/KeyInfoFactory.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeyInfoFactory.java,v 1.12 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import java.math.BigInteger;
import java.security.KeyException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Provider;
import java.security.PublicKey;
import java.security.Security;
import java.security.cert.X509CRL;
import java.util.List;
import javax.xml.crypto.MarshalException;
import javax.xml.crypto.NoSuchMechanismException;
import javax.xml.crypto.URIDereferencer;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dom.DOMStructure;
import javax.xml.crypto.dsig.*;

import sun.security.jca.*;
import sun.security.jca.GetInstance.Instance;

/**
 * A factory for creating {@link KeyInfo} objects from scratch or for
 * unmarshalling a <code>KeyInfo</code> object from a corresponding XML
 * representation.
 *
 * <p>Each instance of <code>KeyInfoFactory</code> supports a specific
 * XML mechanism type. To create a <code>KeyInfoFactory</code>, call one of the
 * static {@link #getInstance getInstance} methods, passing in the XML
 * mechanism type desired, for example:
 *
 * <blockquote><code>
 *   KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 * </code></blockquote>
 *
 * <p>The objects that this factory produces will be based
 * on DOM and abide by the DOM interoperability requirements as defined in the
 * <a href="../../../../../../technotes/guides/security/xmldsig/overview.html#DOM Mechanism
     Requirements">
 * DOM Mechanism Requirements</a> section of the API overview. See the
 * <a href="../../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider">
```

```
 * Service Providers</a> section of the API overview for a list of standard
 * mechanism types.
 *
 * <p><code>KeyInfoFactory</code> implementations are registered and loaded
 * using the {@link java.security.Provider} mechanism.
 * For example, a service provider that supports the
 * DOM mechanism would be specified in the <code>Provider</code> subclass as:
 * <pre>
 *     put("KeyInfoFactory.DOM", "org.example.DOMKeyInfoFactory");
 * </pre>
 *
 * <p>Also, the <code>XMLStructure</code>s that are created by this factory
 * may contain state specific to the <code>KeyInfo</code> and are not
 * intended to be reusable.
 *
 * <p>An implementation MUST minimally support the default mechanism type: DOM.
 *
 * <p>Note that a caller must use the same <code>KeyInfoFactory</code>
 * instance to create the <code>XMLStructure</code>s of a particular
 * <code>KeyInfo</code> object. The behavior is undefined if
 * <code>XMLStructure</code>s from different providers or different mechanism
 * types are used together.
 *
 * <p><b>Concurrent Access</b>
 * <p>The static methods of this class are guaranteed to be thread-safe.
 * Multiple threads may concurrently invoke the static methods defined in this
 * class with no ill effects.
 *
 * <p>However, this is not true for the non-static methods defined by this
 * class. Unless otherwise documented by a specific provider, threads that
 * need to access a single <code>KeyInfoFactory</code> instance concurrently
 * should synchronize amongst themselves and provide the necessary locking.
 * Multiple threads each manipulating a different <code>KeyInfoFactory</code>
 * instance need not synchronize.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 */
public abstract class KeyInfoFactory {

    private String mechanismType;
    private Provider provider;

    /**
     * Default constructor, for invocation by subclasses.
     */
    protected KeyInfoFactory() {}

    /**
     * Returns a <code>KeyInfoFactory</code> that supports the
     * specified XML processing mechanism and representation type (ex: "DOM").
     *
```

```
     * <p>This method uses the standard JCA provider lookup mechanism to
     * locate and instantiate a <code>KeyInfoFactory</code> implementation of
     * the desired mechanism type. It traverses the list of registered security
     * <code>Provider</code>s, starting with the most preferred
     * <code>Provider</code>. A new <code>KeyInfoFactory</code> object
     * from the first <code>Provider</code> that supports the specified
     * mechanism is returned.
     *
     * <p> Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider
        ">
     *    Service Providers</a> section of the API overview for a list of
     *    standard mechanism types.
     * @return a new <code>KeyInfoFactory</code>
     * @throws NullPointerException if <code>mechanismType</code> is
     *    <code>null</code>
     * @throws NoSuchMechanismException if no <code>Provider</code> supports a
     *    <code>KeyInfoFactory</code> implementation for the specified mechanism
     * @see Provider
     */
    public static KeyInfoFactory getInstance(String mechanismType) {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        }
        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("KeyInfoFactory", null, mechanismType);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        KeyInfoFactory factory = (KeyInfoFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns a <code>KeyInfoFactory</code> that supports the
     * requested XML processing mechanism and representation type (ex: "DOM"),
     * as supplied by the specified provider. Note that the specified
     * <code>Provider</code> object does not have to be registered in the
     * provider list.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider
        ">
     *    Service Providers</a> section of the API overview for a list of
```

```
     *    standard mechanism types.
     * @param provider the <code>Provider</code> object
     * @return a new <code>KeyInfoFactory</code>
     * @throws NullPointerException if <code>mechanismType</code> or
     *    <code>provider</code> are <code>null</code>
     * @throws NoSuchMechanismException if a <code>KeyInfoFactory</code>
     *    implementation for the specified mechanism is not available from the
     *    specified <code>Provider</code> object
     * @see Provider
     */
    public static KeyInfoFactory getInstance(String mechanismType,
        Provider provider) {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        } else if (provider == null) {
            throw new NullPointerException("provider cannot be null");
        }

        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("KeyInfoFactory", null, mechanismType, provider);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        KeyInfoFactory factory = (KeyInfoFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns a <code>KeyInfoFactory</code> that supports the
     * requested XML processing mechanism and representation type (ex: "DOM"),
     * as supplied by the specified provider. The specified provider must be
     * registered in the security provider list.
     *
     * <p>Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @param mechanismType the type of the XML processing mechanism and
     *    representation. See the <a
     *    href="../../../../../../technotes/guides/security/xmldsig/overview.html#Service Provider
        ">
     *    Service Providers</a> section of the API overview for a list of
     *    standard mechanism types.
     * @param provider the string name of the provider
     * @return a new <code>KeyInfoFactory</code>
     * @throws NoSuchProviderException if the specified provider is not
     *    registered in the security provider list
     * @throws NullPointerException if <code>mechanismType</code> or
     *    <code>provider</code> are <code>null</code>
     * @throws NoSuchMechanismException if a <code>KeyInfoFactory</code>
```

```
     *    implementation for the specified mechanism is not available from the
     *    specified provider
     * @see Provider
     */
    public static KeyInfoFactory getInstance(String mechanismType,
        String provider) throws NoSuchProviderException {
        if (mechanismType == null) {
            throw new NullPointerException("mechanismType cannot be null");
        } else if (provider == null) {
            throw new NullPointerException("provider cannot be null");
        } else if (provider.length() == 0) {
            throw new NoSuchProviderException();
        }

        Instance instance;
        try {
            instance = GetInstance.getInstance
                ("KeyInfoFactory", null, mechanismType, provider);
        } catch (NoSuchAlgorithmException nsae) {
            throw new NoSuchMechanismException(nsae);
        }
        KeyInfoFactory factory = (KeyInfoFactory) instance.impl;
        factory.mechanismType = mechanismType;
        factory.provider = instance.provider;
        return factory;
    }

    /**
     * Returns a <code>KeyInfoFactory</code> that supports the
     * default XML processing mechanism and representation type ("DOM").
     *
     * <p>This method uses the standard JCA provider lookup mechanism to
     * locate and instantiate a <code>KeyInfoFactory</code> implementation of
     * the default mechanism type. It traverses the list of registered security
     * <code>Provider</code>s, starting with the most preferred
     * <code>Provider</code>.  A new <code>KeyInfoFactory</code> object
     * from the first <code>Provider</code> that supports the DOM mechanism is
     * returned.
     *
     * <p> Note that the list of registered providers may be retrieved via
     * the {@link Security#getProviders() Security.getProviders()} method.
     *
     * @return a new <code>KeyInfoFactory</code>
     * @throws NoSuchMechanismException if no <code>Provider</code> supports a
     *    <code>KeyInfoFactory</code> implementation for the DOM mechanism
     * @see Provider
     */
    public static KeyInfoFactory getInstance() {
        return getInstance("DOM");
    }

    /**
     * Returns the type of the XML processing mechanism and representation
```

```
     * supported by this <code>KeyInfoFactory</code> (ex: "DOM")
     *
     * @return the XML processing mechanism type supported by this
     *    <code>KeyInfoFactory</code>
     */
    public final String getMechanismType() {
        return mechanismType;
    }

    /**
     * Returns the provider of this <code>KeyInfoFactory</code>.
     *
     * @return the provider of this <code>KeyInfoFactory</code>
     */
    public final Provider getProvider() {
        return provider;
    }

    /**
     * Creates a <code>KeyInfo</code> containing the specified list of
     * key information types.
     *
     * @param content a list of one or more {@link XMLStructure}s representing
     *    key information types. The list is defensively copied to protect
     *    against subsequent modification.
     * @return a <code>KeyInfo</code>
     * @throws NullPointerException if <code>content</code> is <code>null</code>
     * @throws IllegalArgumentException if <code>content</code> is empty
     * @throws ClassCastException if <code>content</code> contains any entries
     *    that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract KeyInfo newKeyInfo(List content);

    /**
     * Creates a <code>KeyInfo</code> containing the specified list of key
     * information types and optional id. The
     * <code>id</code> parameter represents the value of an XML
     * <code>ID</code> attribute and is useful for referencing
     * the <code>KeyInfo</code> from other XML structures.
     *
     * @param content a list of one or more {@link XMLStructure}s representing
     *    key information types. The list is defensively copied to protect
     *    against subsequent modification.
     * @param id the value of an XML <code>ID</code> (may be <code>null</code>)
     * @return a <code>KeyInfo</code>
     * @throws NullPointerException if <code>content</code> is <code>null</code>
     * @throws IllegalArgumentException if <code>content</code> is empty
     * @throws ClassCastException if <code>content</code> contains any entries
     *    that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract KeyInfo newKeyInfo(List content, String id);
```

```java

    /**
     * Creates a <code>KeyName</code> from the specified name.
     *
     * @param name the name that identifies the key
     * @return a <code>KeyName</code>
     * @throws NullPointerException if <code>name</code> is <code>null</code>
     */
    public abstract KeyName newKeyName(String name);

    /**
     * Creates a <code>KeyValue</code> from the specified public key.
     *
     * @param key the public key
     * @return a <code>KeyValue</code>
     * @throws KeyException if the <code>key</code>'s algorithm is not
     *    recognized or supported by this <code>KeyInfoFactory</code>
     * @throws NullPointerException if <code>key</code> is <code>null</code>
     */
    public abstract KeyValue newKeyValue(PublicKey key) throws KeyException;

    /**
     * Creates a <code>PGPData</code> from the specified PGP public key
     * identifier.
     *
     * @param keyId a PGP public key identifier as defined in <a href=
     *    "http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>, section 11.2.
     *    The array is cloned to protect against subsequent modification.
     * @return a <code>PGPData</code>
     * @throws NullPointerException if <code>keyId</code> is <code>null</code>
     * @throws IllegalArgumentException if the key id is not in the correct
     *    format
     */
    public abstract PGPData newPGPData(byte[] keyId);

    /**
     * Creates a <code>PGPData</code> from the specified PGP public key
     * identifier, and optional key material packet and list of external
     * elements.
     *
     * @param keyId a PGP public key identifier as defined in <a href=
     *    "http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>, section 11.2.
     *    The array is cloned to protect against subsequent modification.
     * @param keyPacket a PGP key material packet as defined in <a href=
     *    "http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>, section 5.5.
     *    The array is cloned to protect against subsequent modification. May
     *    be <code>null</code>.
     * @param other a list of {@link XMLStructure}s representing elements from
     *    an external namespace. The list is defensively copied to protect
     *    against subsequent modification. May be <code>null</code> or empty.
     * @return a <code>PGPData</code>
     * @throws NullPointerException if <code>keyId</code> is <code>null</code>
     * @throws IllegalArgumentException if the <code>keyId</code> or
```

```
     *    <code>keyPacket</code> is not in the correct format. For
     *    <code>keyPacket</code>, the format of the packet header is
     *    checked and the tag is verified that it is of type key material. The
     *    contents and format of the packet body are not checked.
     * @throws ClassCastException if <code>other</code> contains any
     *    entries that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract PGPData newPGPData(byte[] keyId, byte[] keyPacket,
        List other);

    /**
     * Creates a <code>PGPData</code> from the specified PGP key material
     * packet and optional list of external elements.
     *
     * @param keyPacket a PGP key material packet as defined in <a href=
     *    "http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>, section 5.5.
     *    The array is cloned to protect against subsequent modification.
     * @param other a list of {@link XMLStructure}s representing elements from
     *    an external namespace. The list is defensively copied to protect
     *    against subsequent modification. May be <code>null</code> or empty.
     * @return a <code>PGPData</code>
     * @throws NullPointerException if <code>keyPacket</code> is
     *    <code>null</code>
     * @throws IllegalArgumentException if <code>keyPacket</code> is not in the
     *    correct format. For <code>keyPacket</code>, the format of the packet
     *    header is checked and the tag is verified that it is of type key
     *    material. The contents and format of the packet body are not checked.
     * @throws ClassCastException if <code>other</code> contains any
     *    entries that are not of type {@link XMLStructure}
     */
    @SuppressWarnings("rawtypes")
    public abstract PGPData newPGPData(byte[] keyPacket, List other);

    /**
     * Creates a <code>RetrievalMethod</code> from the specified URI.
     *
     * @param uri the URI that identifies the <code>KeyInfo</code> information
     *    to be retrieved
     * @return a <code>RetrievalMethod</code>
     * @throws NullPointerException if <code>uri</code> is <code>null</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant
     */
    public abstract RetrievalMethod newRetrievalMethod(String uri);

    /**
     * Creates a <code>RetrievalMethod</code> from the specified parameters.
     *
     * @param uri the URI that identifies the <code>KeyInfo</code> information
     *    to be retrieved
     * @param type a URI that identifies the type of <code>KeyInfo</code>
     *    information to be retrieved (may be <code>null</code>)
```

```
     * @param transforms a list of {@link Transform}s. The list is defensively
     *    copied to protect against subsequent modification. May be
     *    <code>null</code> or empty.
     * @return a <code>RetrievalMethod</code>
     * @throws NullPointerException if <code>uri</code> is <code>null</code>
     * @throws IllegalArgumentException if <code>uri</code> is not RFC 2396
     *    compliant
     * @throws ClassCastException if <code>transforms</code> contains any
     *    entries that are not of type {@link Transform}
     */
    @SuppressWarnings("rawtypes")
    public abstract RetrievalMethod newRetrievalMethod(String uri, String type,
        List transforms);

    /**
     * Creates a <code>X509Data</code> containing the specified list of
     * X.509 content.
     *
     * @param content a list of one or more X.509 content types. Valid types are
     *    {@link String} (subject names), <code>byte[]</code> (subject key ids),
     *    {@link java.security.cert.X509Certificate}, {@link X509CRL},
     *    or {@link XMLStructure} ({@link X509IssuerSerial}
     *    objects or elements from an external namespace). Subject names are
     *    distinguished names in RFC 2253 String format. Implementations MUST
     *    support the attribute type keywords defined in RFC 2253 (CN, L, ST,
     *    O, OU, C, STREET, DC and UID). Implementations MAY support additional
     *    keywords. The list is defensively copied to protect against
     *    subsequent modification.
     * @return a <code>X509Data</code>
     * @throws NullPointerException if <code>content</code> is <code>null</code>
     * @throws IllegalArgumentException if <code>content</code> is empty, or
     *    if a subject name is not RFC 2253 compliant or one of the attribute
     *    type keywords is not recognized.
     * @throws ClassCastException if <code>content</code> contains any entries
     *    that are not of one of the valid types mentioned above
     */
    @SuppressWarnings("rawtypes")
    public abstract X509Data newX509Data(List content);

    /**
     * Creates an <code>X509IssuerSerial</code> from the specified X.500 issuer
     * distinguished name and serial number.
     *
     * @param issuerName the issuer's distinguished name in RFC 2253 String
     *    format. Implementations MUST support the attribute type keywords
     *    defined in RFC 2253 (CN, L, ST, O, OU, C, STREET, DC and UID).
     *    Implementations MAY support additional keywords.
     * @param serialNumber the serial number
     * @return an <code>X509IssuerSerial</code>
     * @throws NullPointerException if <code>issuerName</code> or
     *    <code>serialNumber</code> are <code>null</code>
     * @throws IllegalArgumentException if the issuer name is not RFC 2253
     *    compliant or one of the attribute type keywords is not recognized.
```

```
     */
    public abstract X509IssuerSerial newX509IssuerSerial
        (String issuerName, BigInteger serialNumber);

    /**
     * Indicates whether a specified feature is supported.
     *
     * @param feature the feature name (as an absolute URI)
     * @return <code>true</code> if the specified feature is supported,
     *    <code>false</code> otherwise
     * @throws NullPointerException if <code>feature</code> is <code>null</code>
     */
    public abstract boolean isFeatureSupported(String feature);

    /**
     * Returns a reference to the <code>URIDereferencer</code> that is used by
     * default to dereference URIs in {@link RetrievalMethod} objects.
     *
     * @return a reference to the default <code>URIDereferencer</code>
     */
    public abstract URIDereferencer getURIDereferencer();

    /**
     * Unmarshals a new <code>KeyInfo</code> instance from a
     * mechanism-specific <code>XMLStructure</code> (ex: {@link DOMStructure})
     * instance.
     *
     * @param xmlStructure a mechanism-specific XML structure from which to
     *   unmarshal the keyinfo from
     * @return the <code>KeyInfo</code>
     * @throws NullPointerException if <code>xmlStructure</code> is
     *   <code>null</code>
     * @throws ClassCastException if the type of <code>xmlStructure</code> is
     *   inappropriate for this factory
     * @throws MarshalException if an unrecoverable exception occurs during
     *   unmarshalling
     */
    public abstract KeyInfo unmarshalKeyInfo(XMLStructure xmlStructure)
        throws MarshalException;
}
```

12.3 KeyName.java

Secuencia 129: javax/xml/crypto/dsig/keyinfo/KeyName.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeyName.java,v 1.4 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>KeyName</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * A <code>KeyName</code> object contains a string value which may be used
 * by the signer to communicate a key identifier to the recipient. The
 * XML Schema Definition is defined as:
 *
 * <pre>
 * &lt;element name="KeyName" type="string"/&gt;
 * </pre>
 *
 * A <code>KeyName</code> instance may be created by invoking the
 * {@link KeyInfoFactory#newKeyName newKeyName} method of the
 * {@link KeyInfoFactory} class, and passing it a <code>String</code>
 * representing the name of the key; for example:
 * <pre>
 * KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 * KeyName keyName = factory.newKeyName("Alice");
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newKeyName(String)
 */
public interface KeyName extends XMLStructure {

    /**
     * Returns the name of this <code>KeyName</code>.
     *
```

```
     * @return the name of this <code>KeyName</code> (never
     *    <code>null</code>)
     */
    String getName();
}
```

12.4 KeyValue.java

Secuencia 130: javax/xml/crypto/dsig/keyinfo/KeyValue.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: KeyValue.java,v 1.4 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import java.security.KeyException;
import java.security.KeyStore;
import java.security.PublicKey;
import java.security.interfaces.DSAPublicKey;
import java.security.interfaces.RSAPublicKey;
import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>KeyValue</code> element as defined
 * in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>. A
 * <code>KeyValue</code> object contains a single public key that may be
 * useful in validating the signature. The XML schema definition is defined as:
 *
 * <pre>
```

```
*    &lt;element name="KeyValue" type="ds:KeyValueType"/&gt;
*    &lt;complexType name="KeyValueType" mixed="true"&gt;
*      &lt;choice&gt;
*        &lt;element ref="ds:DSAKeyValue"/&gt;
*        &lt;element ref="ds:RSAKeyValue"/&gt;
*        &lt;any namespace="##other" processContents="lax"/&gt;
*      &lt;/choice&gt;
*    &lt;/complexType&gt;
*
*    &lt;element name="DSAKeyValue" type="ds:DSAKeyValueType"/&gt;
*    &lt;complexType name="DSAKeyValueType"&gt;
*      &lt;sequence&gt;
*        &lt;sequence minOccurs="0"&gt;
*          &lt;element name="P" type="ds:CryptoBinary"/&gt;
*          &lt;element name="Q" type="ds:CryptoBinary"/&gt;
*        &lt;/sequence&gt;
*        &lt;element name="G" type="ds:CryptoBinary" minOccurs="0"/&gt;
*        &lt;element name="Y" type="ds:CryptoBinary"/&gt;
*        &lt;element name="J" type="ds:CryptoBinary" minOccurs="0"/&gt;
*        &lt;sequence minOccurs="0"&gt;
*          &lt;element name="Seed" type="ds:CryptoBinary"/&gt;
*          &lt;element name="PgenCounter" type="ds:CryptoBinary"/&gt;
*        &lt;/sequence&gt;
*      &lt;/sequence&gt;
*    &lt;/complexType&gt;
*
*    &lt;element name="RSAKeyValue" type="ds:RSAKeyValueType"/&gt;
*    &lt;complexType name="RSAKeyValueType"&gt;
*      &lt;sequence&gt;
*        &lt;element name="Modulus" type="ds:CryptoBinary"/&gt;
*        &lt;element name="Exponent" type="ds:CryptoBinary"/&gt;
*      &lt;/sequence&gt;
*    &lt;/complexType&gt;
* </pre>
* A <code>KeyValue</code> instance may be created by invoking the
* {@link KeyInfoFactory#newKeyValue newKeyValue} method of the
* {@link KeyInfoFactory} class, and passing it a {@link
* java.security.PublicKey} representing the value of the public key. Here is
* an example of creating a <code>KeyValue</code> from a {@link DSAPublicKey}
* of a {@link java.security.cert.Certificate} stored in a
* {@link java.security.KeyStore}:
* <pre>
* KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
* PublicKey dsaPublicKey = keyStore.getCertificate("myDSASigningCert").getPublicKey();
* KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
* KeyValue keyValue = factory.newKeyValue(dsaPublicKey);
* </pre>
*
* This class returns the <code>DSAKeyValue</code> and
* <code>RSAKeyValue</code> elements as objects of type
* {@link DSAPublicKey} and {@link RSAPublicKey}, respectively. Note that not
* all of the fields in the schema are accessible as parameters of these
* types.
```

```
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newKeyValue(PublicKey)
 */
public interface KeyValue extends XMLStructure {

    /**
     * URI identifying the DSA KeyValue KeyInfo type:
     * http://www.w3.org/2000/09/xmldsig#DSAKeyValue. This can be specified as
     * the value of the <code>type</code> parameter of the
     * {@link RetrievalMethod} class to describe a remote
     * <code>DSAKeyValue</code> structure.
     */
    final static String DSA_TYPE =
        "http://www.w3.org/2000/09/xmldsig#DSAKeyValue";

    /**
     * URI identifying the RSA KeyValue KeyInfo type:
     * http://www.w3.org/2000/09/xmldsig#RSAKeyValue. This can be specified as
     * the value of the <code>type</code> parameter of the
     * {@link RetrievalMethod} class to describe a remote
     * <code>RSAKeyValue</code> structure.
     */
    final static String RSA_TYPE =
        "http://www.w3.org/2000/09/xmldsig#RSAKeyValue";

    /**
     * Returns the public key of this <code>KeyValue</code>.
     *
     * @return the public key of this <code>KeyValue</code>
     * @throws KeyException if this <code>KeyValue</code> cannot be converted
     *    to a <code>PublicKey</code>
     */
    PublicKey getPublicKey() throws KeyException;
}
```

12.5 PGPData.java

Secuencia 131: javax/xml/crypto/dsig/keyinfo/PGPData.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: PGPData.java,v 1.4 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import java.util.Collections;
import java.util.List;
import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>PGPData</code> element as defined in
 * the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>. A
 * <code>PGPData</code> object is used to convey information related to
 * PGP public key pairs and signatures on such keys. The XML Schema Definition
 * is defined as:
 *
 * <pre>
 *    &lt;element name="PGPData" type="ds:PGPDataType"/&gt;
 *    &lt;complexType name="PGPDataType"&gt;
 *      &lt;choice&gt;
 *        &lt;sequence&gt;
 *          &lt;element name="PGPKeyID" type="base64Binary"/&gt;
 *          &lt;element name="PGPKeyPacket" type="base64Binary" minOccurs="0"/&gt;
 *          &lt;any namespace="##other" processContents="lax" minOccurs="0"
 *           maxOccurs="unbounded"/&gt;
 *        &lt;/sequence&gt;
 *        &lt;sequence&gt;
 *          &lt;element name="PGPKeyPacket" type="base64Binary"/&gt;
 *          &lt;any namespace="##other" processContents="lax" minOccurs="0"
 *           maxOccurs="unbounded"/&gt;
 *        &lt;/sequence&gt;
 *      &lt;/choice&gt;
 *    &lt;/complexType&gt;
 * </pre>
 *
 * A <code>PGPData</code> instance may be created by invoking one of the
 * {@link KeyInfoFactory#newPGPData newPGPData} methods of the {@link
 * KeyInfoFactory} class, and passing it
 * <code>byte</code> arrays representing the contents of the PGP public key
 * identifier and/or PGP key material packet, and an optional list of
```

```
 * elements from an external namespace.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newPGPData(byte[])
 * @see KeyInfoFactory#newPGPData(byte[], byte[], List)
 * @see KeyInfoFactory#newPGPData(byte[], List)
 */
public interface PGPData extends XMLStructure {

    /**
     * URI identifying the PGPData KeyInfo type:
     * http://www.w3.org/2000/09/xmldsig#PGPData. This can be specified as the
     * value of the <code>type</code> parameter of the {@link RetrievalMethod}
     * class to describe a remote <code>PGPData</code> structure.
     */
    final static String TYPE = "http://www.w3.org/2000/09/xmldsig#PGPData";

    /**
     * Returns the PGP public key identifier of this <code>PGPData</code> as
     * defined in <a href="http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>,
     * section 11.2.
     *
     * @return the PGP public key identifier (may be <code>null</code> if
     *    not specified). Each invocation of this method returns a new clone
     *    to protect against subsequent modification.
     */
    byte[] getKeyId();

    /**
     * Returns the PGP key material packet of this <code>PGPData</code> as
     * defined in <a href="http://www.ietf.org/rfc/rfc2440.txt">RFC 2440</a>,
     * section 5.5.
     *
     * @return the PGP key material packet (may be <code>null</code> if not
     *    specified). Each invocation of this method returns a new clone to
     *    protect against subsequent modification.
     */
    byte[] getKeyPacket();

    /**
     * Returns an {@link Collections#unmodifiableList unmodifiable list}
     * of {@link XMLStructure}s representing elements from an external
     * namespace.
     *
     * @return an unmodifiable list of <code>XMLStructure</code>s (may be
     *    empty, but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    List getExternalElements();
}
```

12.6 RetrievalMethod.java

Secuencia 132: javax/xml/crypto/dsig/keyinfo/RetrievalMethod.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: RetrievalMethod.java,v 1.8 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import javax.xml.crypto.Data;
import javax.xml.crypto.URIReference;
import javax.xml.crypto.URIReferenceException;
import javax.xml.crypto.XMLCryptoContext;
import javax.xml.crypto.XMLStructure;
import javax.xml.crypto.dsig.Transform;
import java.util.List;

/**
 * A representation of the XML <code>RetrievalMethod</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * A <code>RetrievalMethod</code> object is used to convey a reference to
 * <code>KeyInfo</code> information that is stored at another location.
 * The XML schema definition is defined as:
 *
 * <pre>
 *   &lt;element name="RetrievalMethod" type="ds:RetrievalMethodType"/&gt;
 *   &lt;complexType name="RetrievalMethodType"&gt;
 *     &lt;sequence&gt;
 *       &lt;element name="Transforms" type="ds:TransformsType" minOccurs="0"/&gt;
```

```
 *     &lt;/sequence&gt;
 *     &lt;attribute name="URI" type="anyURI"/&gt;
 *     &lt;attribute name="Type" type="anyURI" use="optional"/&gt;
 *   &lt;/complexType&gt;
 * </pre>
 *
 * A <code>RetrievalMethod</code> instance may be created by invoking one of the
 * {@link KeyInfoFactory#newRetrievalMethod newRetrievalMethod} methods
 * of the {@link KeyInfoFactory} class, and passing it the URI
 * identifying the location of the KeyInfo, an optional type URI identifying
 * the type of KeyInfo, and an optional list of {@link Transform}s; for example:
 * <pre>
 *   KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 *   RetrievalMethod rm = factory.newRetrievalMethod
 *      ("#KeyValue-1", KeyValue.DSA_TYPE, Collections.singletonList(Transform.BASE64));
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newRetrievalMethod(String)
 * @see KeyInfoFactory#newRetrievalMethod(String, String, List)
 */
public interface RetrievalMethod extends URIReference, XMLStructure {

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of {@link Transform}s of this <code>RetrievalMethod</code>.
     *
     * @return an unmodifiable list of <code>Transform</code> objects (may be
     *    empty but never <code>null</code>).
     */
    @SuppressWarnings("rawtypes")
    List getTransforms();

    /**
     * Returns the URI of the referenced <code>KeyInfo</code> information.
     *
     * @return the URI of the referenced <code>KeyInfo</code> information in
     *    RFC 2396 format (never <code>null</code>)
     */
    String getURI();

   /**
    * Dereferences the <code>KeyInfo</code> information referenced by this
    * <code>RetrievalMethod</code> and applies the specified
    * <code>Transform</code>s.
    *
    * @param context an <code>XMLCryptoContext</code> that may contain
    *    additional useful information for dereferencing the URI. The
    *    context's <code>baseURI</code> and <code>dereferencer</code>
    *    parameters (if specified) are used to resolve and dereference this
    *    <code>RetrievalMethod</code>
```

```
    * @return a <code>Data</code> object representing the raw contents of the
    *    <code>KeyInfo</code> information referenced by this
    *    <code>RetrievalMethod</code>. It is the caller's responsibility to
    *    convert the returned data to an appropriate
    *    <code>KeyInfo</code> object.
    * @throws NullPointerException if <code>context</code> is <code>null</code>
    * @throws URIReferenceException if there is an error while dereferencing
    */
    Data dereference(XMLCryptoContext context) throws URIReferenceException;
}
```

12.7 X509Data.java

Secuencia 133: javax/xml/crypto/dsig/keyinfo/X509Data.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: X509Data.java,v 1.4 2005/05/10 16:35:35 mullan Exp $
 */
package javax.xml.crypto.dsig.keyinfo;

import javax.xml.crypto.XMLStructure;
import java.security.cert.X509CRL;
import java.util.List;

/**
 * A representation of the XML <code>X509Data</code> element as defined in
 * the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>. An
 * <code>X509Data</code> object contains one or more identifers of keys
 * or X.509 certificates (or certificates' identifiers or a revocation list).
```

```
 * The XML Schema Definition is defined as:
 *
 * <pre>
 *    &lt;element name="X509Data" type="ds:X509DataType"/&gt;
 *    &lt;complexType name="X509DataType"&gt;
 *        &lt;sequence maxOccurs="unbounded"&gt;
 *          &lt;choice&gt;
 *            &lt;element name="X509IssuerSerial" type="ds:X509IssuerSerialType"/&gt;
 *            &lt;element name="X509SKI" type="base64Binary"/&gt;
 *            &lt;element name="X509SubjectName" type="string"/&gt;
 *            &lt;element name="X509Certificate" type="base64Binary"/&gt;
 *            &lt;element name="X509CRL" type="base64Binary"/&gt;
 *            &lt;any namespace="##other" processContents="lax"/&gt;
 *          &lt;/choice&gt;
 *        &lt;/sequence&gt;
 *    &lt;/complexType&gt;
 *
 *    &lt;complexType name="X509IssuerSerialType"&gt;
 *      &lt;sequence&gt;
 *        &lt;element name="X509IssuerName" type="string"/&gt;
 *        &lt;element name="X509SerialNumber" type="integer"/&gt;
 *      &lt;/sequence&gt;
 *    &lt;/complexType&gt;
 * </pre>
 *
 * An <code>X509Data</code> instance may be created by invoking the
 * {@link KeyInfoFactory#newX509Data newX509Data} methods of the
 * {@link KeyInfoFactory} class and passing it a list of one or more
 * {@link XMLStructure}s representing X.509 content; for example:
 * <pre>
 *   KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 *   X509Data x509Data = factory.newX509Data
 *       (Collections.singletonList("cn=Alice"));
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see KeyInfoFactory#newX509Data(List)
 */
//@@@ check for illegal combinations of data violating MUSTs in W3c spec
public interface X509Data extends XMLStructure {

    /**
     * URI identifying the X509Data KeyInfo type:
     * http://www.w3.org/2000/09/xmldsig#X509Data. This can be specified as
     * the value of the <code>type</code> parameter of the
     * {@link RetrievalMethod} class to describe a remote
     * <code>X509Data</code> structure.
     */
    final static String TYPE = "http://www.w3.org/2000/09/xmldsig#X509Data";

    /**
```

```
     * URI identifying the binary (ASN.1 DER) X.509 Certificate KeyInfo type:
     * http://www.w3.org/2000/09/xmldsig#rawX509Certificate. This can be
     * specified as the value of the <code>type</code> parameter of the
     * {@link RetrievalMethod} class to describe a remote X509 Certificate.
     */
    final static String RAW_X509_CERTIFICATE_TYPE =
        "http://www.w3.org/2000/09/xmldsig#rawX509Certificate";

    /**
     * Returns an {@link java.util.Collections#unmodifiableList unmodifiable
     * list} of the content in this <code>X509Data</code>. Valid types are
     * {@link String} (subject names), <code>byte[]</code> (subject key ids),
     * {@link java.security.cert.X509Certificate}, {@link X509CRL},
     * or {@link XMLStructure} ({@link X509IssuerSerial}
     * objects or elements from an external namespace).
     *
     * @return an unmodifiable list of the content in this <code>X509Data</code>
     *    (never <code>null</code> or empty)
     */
    @SuppressWarnings("rawtypes")
    List getContent();
}
```

12.8 X509IssuerSerial.java

Secuencia 134: javax/xml/crypto/dsig/keyinfo/X509IssuerSerial.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: X509IssuerSerial.java,v 1.4 2005/05/10 16:35:35 mullan Exp $
 */
```

```java
package javax.xml.crypto.dsig.keyinfo;

import java.math.BigInteger;
import java.security.cert.X509Certificate;
import javax.xml.crypto.XMLStructure;

/**
 * A representation of the XML <code>X509IssuerSerial</code> element as
 * defined in the <a href="http://www.w3.org/TR/xmldsig-core/">
 * W3C Recommendation for XML-Signature Syntax and Processing</a>.
 * An <code>X509IssuerSerial</code> object contains an X.509 issuer
 * distinguished name (DN) and serial number pair. The XML schema definition is
 * defined as:
 *
 * <pre>
 *   &lt;element name="X509IssuerSerial" type="ds:X509IssuerSerialType"/&gt;
 *   &lt;complexType name="X509IssuerSerialType"&gt;
 *     &lt;sequence&gt;
 *       &lt;element name="X509IssuerName" type="string"/&gt;
 *       &lt;element name="X509SerialNumber" type="integer"/&gt;
 *     &lt;/sequence&gt;
 *   &lt;/complexType&gt;
 * </pre>
 *
 * An <code>X509IssuerSerial</code> instance may be created by invoking the
 * {@link KeyInfoFactory#newX509IssuerSerial newX509IssuerSerial} method
 * of the {@link KeyInfoFactory} class, and passing it a
 * <code>String</code> and <code>BigInteger</code> representing the X.500
 * DN and serial number. Here is an example of creating an
 * <code>X509IssuerSerial</code> from the issuer DN and serial number of an
 * existing {@link X509Certificate}:
 * <pre>
 * KeyInfoFactory factory = KeyInfoFactory.getInstance("DOM");
 * X509IssuerSerial issuer = factory.newX509IssuerSerial
 *     (cert.getIssuerX500Principal().getName(), cert.getSerialNumber());
 * </pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see X509Data#getContent
 * @see KeyInfoFactory#newX509IssuerSerial(String, BigInteger)
 */
public interface X509IssuerSerial extends XMLStructure {

    /**
     * Returns the X.500 distinguished name of this
     * <code>X509IssuerSerial</code> in
     * <a href="http://www.ietf.org/rfc/rfc2253.txt">RFC 2253</a> String format.
     *
     * @return the X.500 distinguished name in RFC 2253 String format (never
     *    <code>null</code>)
     */
```

```
    String getIssuerName();

    /**
     * Returns the serial number of this <code>X509IssuerSerial</code>.
     *
     * @return the serial number (never <code>null</code>)
     */
    BigInteger getSerialNumber();
}
```

13 Xml Crypto Dsig Spec (javax.xml.crypto.dsig.spec package)

13.1 C14NMethodParameterSpec.java

Secuencia 135: javax/xml/crypto/dsig/spec/C14NMethodParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: C14NMethodParameterSpec.java,v 1.3 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.CanonicalizationMethod;

/**
 * A specification of algorithm parameters for a {@link CanonicalizationMethod}
 * Algorithm. The purpose of this interface is to group (and provide type
 * safety for) all canonicalization (C14N) parameter specifications. All
 * canonicalization parameter specifications must implement this interface.
 *
 * @author Sean Mullan
```

```
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see CanonicalizationMethod
 */
public interface C14NMethodParameterSpec extends TransformParameterSpec {}
```

13.2 DigestMethodParameterSpec.java

Secuencia 136: javax/xml/crypto/dsig/spec/DigestMethodParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: DigestMethodParameterSpec.java,v 1.3 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.DigestMethod;
import java.security.spec.AlgorithmParameterSpec;

/**
 * A specification of algorithm parameters for a {@link DigestMethod}
 * algorithm. The purpose of this interface is to group (and provide type
 * safety for) all digest method parameter specifications. All digest method
 * parameter specifications must implement this interface.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see DigestMethod
 */
public interface DigestMethodParameterSpec extends AlgorithmParameterSpec {}
```

13.3 ExcC14NParameterSpec.java

Secuencia 137: javax/xml/crypto/dsig/spec/ExcC14NParameterSpec.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: ExcC14NParameterSpec.java,v 1.7 2005/05/13 18:45:42 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.CanonicalizationMethod;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Parameters for the W3C Recommendation:
 * <a href="http://www.w3.org/TR/xml-exc-c14n/">
 * Exclusive XML Canonicalization (C14N) algorithm</a>. The
 * parameters include an optional inclusive namespace prefix list. The XML
 * Schema Definition of the Exclusive XML Canonicalization parameters is
 * defined as:
 * <pre><code>
 * &lt;schema xmlns="http://www.w3.org/2001/XMLSchema"
 *         xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
 *         targetNamespace="http://www.w3.org/2001/10/xml-exc-c14n#"
 *         version="0.1" elementFormDefault="qualified"&gt;
 *
 * &lt;element name="InclusiveNamespaces" type="ec:InclusiveNamespaces"/&gt;
 * &lt;complexType name="InclusiveNamespaces"&gt;
 *   &lt;attribute name="PrefixList" type="xsd:string"/&gt;
```

```
 * &lt;/complexType&gt;
 * &lt;/schema&gt;
 * </code></pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see CanonicalizationMethod
 */
public final class ExcC14NParameterSpec implements C14NMethodParameterSpec {

    private List<String> preList;

    /**
     * Indicates the default namespace ("#default").
     */
    public static final String DEFAULT = "#default";

    /**
     * Creates a <code>ExcC14NParameterSpec</code> with an empty prefix
     * list.
     */
    public ExcC14NParameterSpec() {
        preList = Collections.emptyList();
    }

    /**
     * Creates a <code>ExcC14NParameterSpec</code> with the specified list
     * of prefixes. The list is copied to protect against subsequent
     * modification.
     *
     * @param prefixList the inclusive namespace prefix list. Each entry in
     *    the list is a <code>String</code> that represents a namespace prefix.
     * @throws NullPointerException if <code>prefixList</code> is
     *    <code>null</code>
     * @throws ClassCastException if any of the entries in the list are not
     *    of type <code>String</code>
     */
    @SuppressWarnings("rawtypes")
    public ExcC14NParameterSpec(List prefixList) {
        if (prefixList == null) {
            throw new NullPointerException("prefixList cannot be null");
        }
        List<?> copy = new ArrayList<>((List<?>)prefixList);
        for (int i = 0, size = copy.size(); i < size; i++) {
            if (!(copy.get(i) instanceof String)) {
                throw new ClassCastException("not a String");
            }
        }

        @SuppressWarnings("unchecked")
        List<String> temp = (List<String>)copy;

```

```
        preList = Collections.unmodifiableList(temp);
    }

    /**
     * Returns the inclusive namespace prefix list. Each entry in the list
     * is a <code>String</code> that represents a namespace prefix.
     *
     * <p>This implementation returns an {@link
     * java.util.Collections#unmodifiableList unmodifiable list}.
     *
     * @return the inclusive namespace prefix list (may be empty but never
     *    <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    public List getPrefixList() {
        return preList;
    }
}
```

13.4 HMACParameterSpec.java

Secuencia 138: javax/xml/crypto/dsig/spec/HMACParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: HMACParameterSpec.java,v 1.4 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.SignatureMethod;

```

```
/**
 * Parameters for the <a href="http://www.w3.org/TR/xmldsig-core/#sec-MACs">
 * XML Signature HMAC Algorithm</a>. The parameters include an optional output
 * length which specifies the MAC truncation length in bits. The resulting
 * HMAC will be truncated to the specified number of bits. If the parameter is
 * not specified, then this implies that all the bits of the hash are to be
 * output. The XML Schema Definition of the <code>HMACOutputLength</code>
 * element is defined as:
 * <pre><code>
 * &lt;element name="HMACOutputLength" minOccurs="0" type="ds:HMACOutputLengthType"/&gt;
 * &lt;simpleType name="HMACOutputLengthType"&gt;
 *   &lt;restriction base="integer"/&gt;
 * &lt;/simpleType&gt;
 * </code></pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see SignatureMethod
 * @see <a href="http://www.ietf.org/rfc/rfc2104.txt">RFC 2104</a>
 */
public final class HMACParameterSpec implements SignatureMethodParameterSpec {

    private int outputLength;

    /**
     * Creates an <code>HMACParameterSpec</code> with the specified truncation
     * length.
     *
     * @param outputLength the truncation length in number of bits
     */
    public HMACParameterSpec(int outputLength) {
        this.outputLength = outputLength;
    }

    /**
     * Returns the truncation length.
     *
     * @return the truncation length in number of bits
     */
    public int getOutputLength() {
        return outputLength;
    }
}
```

13.5 SignatureMethodParameterSpec.java

Secuencia 139: javax/xml/crypto/dsig/spec/SignatureMethodParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: SignatureMethodParameterSpec.java,v 1.3 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.SignatureMethod;
import java.security.spec.AlgorithmParameterSpec;

/**
 * A specification of algorithm parameters for an XML {@link SignatureMethod}
 * algorithm. The purpose of this interface is to group (and provide type
 * safety for) all signature method parameter specifications. All signature
 * method parameter specifications must implement this interface.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see SignatureMethod
 */
public interface SignatureMethodParameterSpec extends AlgorithmParameterSpec {}
```

13.6 TransformParameterSpec.java

Secuencia 140: javax/xml/crypto/dsig/spec/TransformParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: TransformParameterSpec.java,v 1.3 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.Transform;
import java.security.spec.AlgorithmParameterSpec;

/**
 * A specification of algorithm parameters for a {@link Transform}
 * algorithm. The purpose of this interface is to group (and provide type
 * safety for) all transform parameter specifications. All transform parameter
 * specifications must implement this interface.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see Transform
 */
public interface TransformParameterSpec extends AlgorithmParameterSpec {}
```

13.7 XPathFilter2ParameterSpec.java

Secuencia 141: javax/xml/crypto/dsig/spec/XPathFilter2ParameterSpec.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XPathFilter2ParameterSpec.java,v 1.7 2005/05/13 18:45:42 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import javax.xml.crypto.dsig.Transform;

/**
 * Parameters for the W3C Recommendation
 * <a href="http://www.w3.org/TR/xmldsig-filter2/">
 * XPath Filter 2.0 Transform Algorithm</a>.
 * The parameters include a list of one or more {@link XPathType} objects.
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see Transform
 * @see XPathFilterParameterSpec
 */
public final class XPathFilter2ParameterSpec implements TransformParameterSpec {

    private final List<XPathType> xPathList;

    /**
     * Creates an <code>XPathFilter2ParameterSpec</code>.
     *
     * @param xPathList a list of one or more {@link XPathType} objects. The
     *    list is defensively copied to protect against subsequent modification.
     * @throws ClassCastException if <code>xPathList</code> contains any
     *    entries that are not of type {@link XPathType}
     * @throws IllegalArgumentException if <code>xPathList</code> is empty
     * @throws NullPointerException if <code>xPathList</code> is
     *    <code>null</code>
     */
    @SuppressWarnings("rawtypes")
    public XPathFilter2ParameterSpec(List xPathList) {
        if (xPathList == null) {
            throw new NullPointerException("xPathList cannot be null");
        }
        List<?> xPathListCopy = new ArrayList<>((List<?>)xPathList);
        if (xPathListCopy.isEmpty()) {
```

```
            throw new IllegalArgumentException("xPathList cannot be empty");
        }
        int size = xPathListCopy.size();
        for (int i = 0; i < size; i++) {
            if (!(xPathListCopy.get(i) instanceof XPathType)) {
                throw new ClassCastException
                    ("xPathList["+i+"] is not a valid type");
            }
        }

        @SuppressWarnings("unchecked")
        List<XPathType> temp = (List<XPathType>)xPathListCopy;

        this.xPathList = Collections.unmodifiableList(temp);
    }

    /**
     * Returns a list of one or more {@link XPathType} objects.
     * <p>
     * This implementation returns an {@link Collections#unmodifiableList
     * unmodifiable list}.
     *
     * @return a <code>List</code> of <code>XPathType</code> objects
     *    (never <code>null</code> or empty)
     */
    @SuppressWarnings("rawtypes")
    public List getXPathList() {
        return xPathList;
    }
}
```

13.8 XPathFilterParameterSpec.java

Secuencia 142: javax/xml/crypto/dsig/spec/XPathFilterParameterSpec.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */
/*
 * $Id: XPathFilterParameterSpec.java,v 1.4 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.Transform;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;

/**
 * Parameters for the <a href="http://www.w3.org/TR/xmldsig-core/#sec-XPath">
 * XPath Filtering Transform Algorithm</a>.
 * The parameters include the XPath expression and an optional <code>Map</code>
 * of additional namespace prefix mappings. The XML Schema Definition of
 * the XPath Filtering transform parameters is defined as:
 * <pre><code>
 * &lt;element name="XPath" type="string"/&gt;
 * </code></pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see Transform
 */
public final class XPathFilterParameterSpec implements TransformParameterSpec {

    private String xPath;
    private Map<String,String> nsMap;

    /**
     * Creates an <code>XPathFilterParameterSpec</code> with the specified
     * XPath expression.
     *
     * @param xPath the XPath expression to be evaluated
     * @throws NullPointerException if <code>xPath</code> is <code>null</code>
     */
    public XPathFilterParameterSpec(String xPath) {
        if (xPath == null) {
            throw new NullPointerException();
        }
        this.xPath = xPath;
        this.nsMap = Collections.emptyMap();
    }

    /**
```

```
     * Creates an <code>XPathFilterParameterSpec</code> with the specified
     * XPath expression and namespace map. The map is copied to protect against
     * subsequent modification.
     *
     * @param xPath the XPath expression to be evaluated
     * @param namespaceMap the map of namespace prefixes. Each key is a
     *    namespace prefix <code>String</code> that maps to a corresponding
     *    namespace URI <code>String</code>.
     * @throws NullPointerException if <code>xPath</code> or
     *    <code>namespaceMap</code> are <code>null</code>
     * @throws ClassCastException if any of the map's keys or entries are not
     *    of type <code>String</code>
     */
    @SuppressWarnings("rawtypes")
    public XPathFilterParameterSpec(String xPath, Map namespaceMap) {
        if (xPath == null || namespaceMap == null) {
            throw new NullPointerException();
        }
        this.xPath = xPath;
        Map<?,?> copy = new HashMap<>((Map<?,?>)namespaceMap);
        Iterator<? extends Map.Entry<?,?>> entries = copy.entrySet().iterator();
        while (entries.hasNext()) {
            Map.Entry<?,?> me = entries.next();
            if (!(me.getKey() instanceof String) ||
                !(me.getValue() instanceof String)) {
                throw new ClassCastException("not a String");
            }
        }

        @SuppressWarnings("unchecked")
        Map<String,String> temp = (Map<String,String>)copy;

        nsMap = Collections.unmodifiableMap(temp);
    }

    /**
     * Returns the XPath expression to be evaluated.
     *
     * @return the XPath expression to be evaluated
     */
    public String getXPath() {
        return xPath;
    }

    /**
     * Returns a map of namespace prefixes. Each key is a namespace prefix
     * <code>String</code> that maps to a corresponding namespace URI
     * <code>String</code>.
     * <p>
     * This implementation returns an {@link Collections#unmodifiableMap
     * unmodifiable map}.
     *
     * @return a <code>Map</code> of namespace prefixes to namespace URIs (may
```

```
     *    be empty, but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
    public Map getNamespaceMap() {
        return nsMap;
    }
}
```

13.9 XPathType.java

Secuencia 143: javax/xml/crypto/dsig/spec/XPathType.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XPathType.java,v 1.4 2005/05/10 16:40:17 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import java.util.Collections;
import java.util.Iterator;
import java.util.HashMap;
import java.util.Map;

/**
 * The XML Schema Definition of the <code>XPath</code> element as defined in the
 * <a href="http://www.w3.org/TR/xmldsig-filter2">
 * W3C Recommendation for XML-Signature XPath Filter 2.0</a>:
 * <pre><code>
 * &lt;schema xmlns="http://www.w3.org/2001/XMLSchema"
 *         xmlns:xf="http://www.w3.org/2002/06/xmldsig-filter2"
 *         targetNamespace="http://www.w3.org/2002/06/xmldsig-filter2"
```

```
 *         version="0.1" elementFormDefault="qualified"&gt;
 *
 * &lt;element name="XPath"
 *          type="xf:XPathType"/&gt;
 *
 * &lt;complexType name="XPathType"&gt;
 *   &lt;simpleContent&gt;
 *     &lt;extension base="string"&gt;
 *       &lt;attribute name="Filter"&gt;
 *         &lt;simpleType&gt;
 *           &lt;restriction base="string"&gt;
 *             &lt;enumeration value="intersect"/&gt;
 *             &lt;enumeration value="subtract"/&gt;
 *             &lt;enumeration value="union"/&gt;
 *           &lt;/restriction&gt;
 *         &lt;/simpleType&gt;
 *       &lt;/attribute&gt;
 *     &lt;/extension&gt;
 *   &lt;/simpleContent&gt;
 * &lt;/complexType&gt;
 * </code></pre>
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see XPathFilter2ParameterSpec
 */
public class XPathType {

    /**
     * Represents the filter set operation.
     */
    public static class Filter {
        private final String operation;

        private Filter(String operation) {
            this.operation = operation;
        }

        /**
         * Returns the string form of the operation.
         *
         * @return the string form of the operation
         */
        public String toString() {
            return operation;
        }

        /**
         * The intersect filter operation.
         */
        public static final Filter INTERSECT = new Filter("intersect");

```

```
        /**
         * The subtract filter operation.
         */
        public static final Filter SUBTRACT = new Filter("subtract");

        /**
         * The union filter operation.
         */
        public static final Filter UNION = new Filter("union");
    }

    private final String expression;
    private final Filter filter;
    private Map<String,String> nsMap;

    /**
     * Creates an <code>XPathType</code> instance with the specified XPath
     * expression and filter.
     *
     * @param expression the XPath expression to be evaluated
     * @param filter the filter operation ({@link Filter#INTERSECT},
     *    {@link Filter#SUBTRACT}, or {@link Filter#UNION})
     * @throws NullPointerException if <code>expression</code> or
     *    <code>filter</code> is <code>null</code>
     */
    public XPathType(String expression, Filter filter) {
        if (expression == null) {
            throw new NullPointerException("expression cannot be null");
        }
        if (filter == null) {
            throw new NullPointerException("filter cannot be null");
        }
        this.expression = expression;
        this.filter = filter;
        this.nsMap = Collections.emptyMap();
    }

    /**
     * Creates an <code>XPathType</code> instance with the specified XPath
     * expression, filter, and namespace map. The map is copied to protect
     * against subsequent modification.
     *
     * @param expression the XPath expression to be evaluated
     * @param filter the filter operation ({@link Filter#INTERSECT},
     *    {@link Filter#SUBTRACT}, or {@link Filter#UNION})
     * @param namespaceMap the map of namespace prefixes. Each key is a
     *    namespace prefix <code>String</code> that maps to a corresponding
     *    namespace URI <code>String</code>.
     * @throws NullPointerException if <code>expression</code>,
     *    <code>filter</code> or <code>namespaceMap</code> are
     *    <code>null</code>
     * @throws ClassCastException if any of the map's keys or entries are
     *    not of type <code>String</code>
```

```
     */
    @SuppressWarnings("rawtypes")
    public XPathType(String expression, Filter filter, Map namespaceMap) {
        this(expression, filter);
        if (namespaceMap == null) {
            throw new NullPointerException("namespaceMap cannot be null");
        }
        Map<?,?> copy = new HashMap<>((Map<?,?>)namespaceMap);
        Iterator<? extends Map.Entry<?,?>> entries = copy.entrySet().iterator();
        while (entries.hasNext()) {
            Map.Entry<?,?> me = entries.next();
            if (!(me.getKey() instanceof String) ||
                !(me.getValue() instanceof String)) {
                throw new ClassCastException("not a String");
            }
        }

        @SuppressWarnings("unchecked")
        Map<String,String> temp = (Map<String,String>)copy;

        nsMap = Collections.unmodifiableMap(temp);
    }

    /**
     * Returns the XPath expression to be evaluated.
     *
     * @return the XPath expression to be evaluated
     */
    public String getExpression() {
        return expression;
    }

    /**
     * Returns the filter operation.
     *
     * @return the filter operation
     */
    public Filter getFilter() {
        return filter;
    }

    /**
     * Returns a map of namespace prefixes. Each key is a namespace prefix
     * <code>String</code> that maps to a corresponding namespace URI
     * <code>String</code>.
     * <p>
     * This implementation returns an {@link Collections#unmodifiableMap
     * unmodifiable map}.
     *
     * @return a <code>Map</code> of namespace prefixes to namespace URIs
     *    (may be empty, but never <code>null</code>)
     */
    @SuppressWarnings("rawtypes")
```

```
    public Map getNamespaceMap() {
        return nsMap;
    }
}
```

13.10 XSLTTransformParameterSpec.java

Secuencia 144: javax/xml/crypto/dsig/spec/XSLTTransformParameterSpec.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
/*
 * $Id: XSLTTransformParameterSpec.java,v 1.4 2005/05/10 16:40:18 mullan Exp $
 */
package javax.xml.crypto.dsig.spec;

import javax.xml.crypto.dsig.Transform;
import javax.xml.crypto.XMLStructure;

/**
 * Parameters for the <a href="http://www.w3.org/TR/1999/REC-xslt-19991116">
 * XSLT Transform Algorithm</a>.
 * The parameters include a namespace-qualified stylesheet element.
 *
 * <p>An <code>XSLTTransformParameterSpec</code> is instantiated with a
 * mechanism-dependent (ex: DOM) stylesheet element. For example:
 * <pre>
 *   DOMStructure stylesheet = new DOMStructure(element)
 *   XSLTTransformParameterSpec spec = new XSLTTransformParameterSpec(stylesheet);
 * </pre>
 * where <code>element</code> is an {@link org.w3c.dom.Element} containing
 * the namespace-qualified stylesheet element.
```

```
 *
 * @author Sean Mullan
 * @author JSR 105 Expert Group
 * @since 1.6
 * @see Transform
 */
public final class XSLTTransformParameterSpec implements TransformParameterSpec{
    private XMLStructure stylesheet;

    /**
     * Creates an <code>XSLTTransformParameterSpec</code> with the specified
     * stylesheet.
     *
     * @param stylesheet the XSLT stylesheet to be used
     * @throws NullPointerException if <code>stylesheet</code> is
     *    <code>null</code>
     */
    public XSLTTransformParameterSpec(XMLStructure stylesheet) {
        if (stylesheet == null) {
            throw new NullPointerException();
        }
        this.stylesheet = stylesheet;
    }

    /**
     * Returns the stylesheet.
     *
     * @return the stylesheet
     */
    public XMLStructure getStylesheet() {
        return stylesheet;
    }
}
```

14 Xml Datatype (javax.xml.datatype package)

14.1 DatatypeConfigurationException.java

Secuencia 145: javax/xml/datatype/DatatypeConfigurationException.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

/**
 * <p>Indicates a serious configuration error.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */

public class DatatypeConfigurationException extends Exception {

    /**
     * <p>Create a new <code>DatatypeConfigurationException</code> with
     * no specified detail mesage and cause.</p>
     */

    public DatatypeConfigurationException() {
        super();
    }

    /**
     * <p>Create a new <code>DatatypeConfigurationException</code> with
         * the specified detail message.</p>
     *
         * @param message The detail message.
     */

    public DatatypeConfigurationException(String message) {
        super(message);
    }

        /**
         * <p>Create a new <code>DatatypeConfigurationException</code> with
         * the specified detail message and cause.</p>
         *
         * @param message The detail message.
         * @param cause The cause.  A <code>null</code> value is permitted, and indicates that the
             cause is nonexistent or unknown.
         */

        public DatatypeConfigurationException(String message, Throwable cause) {
                super(message, cause);
```

```
        }

        /**
         * <p>Create a new <code>DatatypeConfigurationException</code> with
         * the specified cause.</p>
         *
         * @param cause The cause.  A <code>null</code> value is permitted, and indicates that the
             cause is nonexistent or unknown.
         */

        public DatatypeConfigurationException(Throwable cause) {
                super(cause);
        }
}
```

14.2 DatatypeConstants.java

Secuencia 146: javax/xml/datatype/DatatypeConstants.java

```
/*
 * Copyright (c) 2004, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import javax.xml.XMLConstants;
import javax.xml.namespace.QName;

/**
 * <p>Utility class to contain basic Datatype values as constants.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
```

```
 */

public final class DatatypeConstants {

    /**
     * <p>Private constructor to prevent instantiation.</p>
     */
        private DatatypeConstants() {
        }

        /**
         * Value for first month of year.
         */
        public static final int JANUARY  = 1;

        /**
         * Value for second month of year.
         */
        public static final int FEBRUARY = 2;

        /**
         * Value for third month of year.
         */
        public static final int MARCH    = 3;

        /**
         * Value for fourth month of year.
         */
        public static final int APRIL    = 4;

        /**
         * Value for fifth month of year.
         */
        public static final int MAY      = 5;

        /**
         * Value for sixth month of year.
         */
        public static final int JUNE     = 6;

        /**
         * Value for seventh month of year.
         */
        public static final int JULY     = 7;

        /**
         * Value for eighth month of year.
         */
        public static final int AUGUST   = 8;

        /**
         * Value for ninth month of year.
         */
```

```java
        public static final int SEPTEMBER = 9;

        /**
         * Value for tenth month of year.
         */
        public static final int OCTOBER = 10;

        /**
         * Value for eleven month of year.
         */
        public static final int NOVEMBER = 11;

        /**
         * Value for twelve month of year.
         */
        public static final int DECEMBER = 12;

        /**
         * <p>Comparison result.</p>
         */
        public static final int LESSER = -1;

        /**
         * <p>Comparison result.</p>
         */
        public static final int EQUAL =  0;

        /**
         * <p>Comparison result.</p>
         */
        public static final int GREATER =  1;

        /**
         * <p>Comparison result.</p>
         */
        public static final int INDETERMINATE =  2;

        /**
         * Designation that an "int" field is not set.
         */
        public static final int FIELD_UNDEFINED = Integer.MIN_VALUE;

        /**
         * <p>A constant that represents the years field.</p>
         */
        public static final Field YEARS = new Field("YEARS", 0);

        /**
         * <p>A constant that represents the months field.</p>
         */
        public static final Field MONTHS = new Field("MONTHS", 1);

        /**
```

```java
         * <p>A constant that represents the days field.</p>
         */
        public static final Field DAYS = new Field("DAYS", 2);

        /**
         * <p>A constant that represents the hours field.</p>
         */
        public static final Field HOURS = new Field("HOURS", 3);

        /**
         * <p>A constant that represents the minutes field.</p>
         */
        public static final Field MINUTES = new Field("MINUTES", 4);

        /**
         * <p>A constant that represents the seconds field.</p>
         */
        public static final Field SECONDS = new Field("SECONDS", 5);

        /**
         * Type-safe enum class that represents six fields
         * of the {@link Duration} class.
         * @since 1.5
         */
        public static final class Field {

                /**
                 * <p><code>String</code> representation of <code>Field</code>.</p>
                 */
                private final String str;
                /**
                 * <p>Unique id of the field.</p>
                 *
                 * <p>This value allows the {@link Duration} class to use switch
                 * statements to process fields.</p>
                 */
                private final int id;

                /**
                 * <p>Construct a <code>Field</code> with specified values.</p>
                 * @param str <code>String</code> representation of <code>Field</code>
                 * @param id  <code>int</code> representation of <code>Field</code>
                 */
                private Field(final String str, final int id) {
                        this.str = str;
                        this.id = id;
                }
                /**
                 * Returns a field name in English. This method
                 * is intended to be used for debugging/diagnosis
                 * and not for display to end-users.
                 *
                 * @return
```

```
                 *      a non-null valid String constant.
                 */
                public String toString() { return str; }

                /**
                 * <p>Get id of this Field.</p>
                 *
                 * @return Id of field.
                 */
                public int getId() {
                        return id;
                }
        }

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>dateTime</code>.</p>
         */
        public static final QName DATETIME = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "
            dateTime");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>time</code>.</p>
         */
        public static final QName TIME = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "time");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>date</code>.</p>
         */
        public static final QName DATE = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "date");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>gYearMonth</code>.</p>
         */
        public static final QName GYEARMONTH = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "
            gYearMonth");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>gMonthDay</code>.</p>
         */
        public static final QName GMONTHDAY = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "
            gMonthDay");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>gYear</code>.</p>
         */
        public static final QName GYEAR = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gYear");

        /**
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>gMonth</code>.</p>
         */
        public static final QName GMONTH = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gMonth");

        /**
```

```
         * <p>Fully qualified name for W3C XML Schema 1.0 datatype <code>gDay</code>.</p>
         */
        public static final QName GDAY = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gDay");

        /**
         * <p>Fully qualified name for W3C XML Schema datatype <code>duration</code>.</p>
         */
        public static final QName DURATION = new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "
            duration");

        /**
         * <p>Fully qualified name for XQuery 1.0 and XPath 2.0 datatype <code>dayTimeDuration</
             code>.</p>
         */
        public static final QName DURATION_DAYTIME = new QName(XMLConstants.
            W3C_XPATH_DATATYPE_NS_URI, "dayTimeDuration");

        /**
         * <p>Fully qualified name for XQuery 1.0 and XPath 2.0 datatype <code>yearMonthDuration</
             code>.</p>
         */
        public static final QName DURATION_YEARMONTH = new QName(XMLConstants.
            W3C_XPATH_DATATYPE_NS_URI, "yearMonthDuration");

        /**
         * W3C XML Schema max timezone offset is -14:00. Zone offset is in minutes.
         */
        public static final int MAX_TIMEZONE_OFFSET = -14 * 60;

        /**
         * W3C XML Schema min timezone offset is +14:00. Zone offset is in minutes.
         */
        public static final int MIN_TIMEZONE_OFFSET = 14 * 60;

}
```

14.3 DatatypeFactory.java

Secuencia 147: javax/xml/datatype/DatatypeFactory.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.GregorianCalendar;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * <p>Factory that creates new <code>javax.xml.datatype</code> <code>Object</code>s that map XML to /from Java <code>Object</code>s.</p>
 *
 * <p>A new instance of the <code>DatatypeFactory</code> is created through the {@link #newInstance()} method
 * that uses the following implementation resolution mechanisms to determine an implementation:</p>
 * <ol>
 *    <li>
 *      If the system property specified by {@link #DATATYPEFACTORY_PROPERTY}, "<code>javax.xml.datatype.DatatypeFactory</code>",
 *      exists, a class with the name of the property value is instantiated.
 *      Any Exception thrown during the instantiation process is wrapped as a {@link DatatypeConfigurationException}.
 *    </li>
 *    <li>
 *      If the file ${JAVA_HOME}/lib/jaxp.properties exists, it is loaded in a {@link java.util.Properties} <code>Object</code>.
 *      The <code>Properties</code> <code>Object </code> is then queried for the property as documented in the prior step
 *      and processed as documented in the prior step.
 *    </li>
 *    <li>
 *     Uses the service-provider loading facilities, defined by the {@link java.util.ServiceLoader} class, to attempt
 *     to locate and load an implementation of the service using the {@linkplain
 *     java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
 *     the service-provider loading facility will use the {@linkplain
 *     java.lang.Thread#getContextClassLoader() current thread's context class loader}
 *     to attempt to load the service. If the context class
 *     loader is null, the {@linkplain
 *     ClassLoader#getSystemClassLoader() system class loader} will be used.
 *     <br>
 *     In case of {@link java.util.ServiceConfigurationError service
```

```
 *     configuration error} a {@link javax.xml.datatype.DatatypeConfigurationException}
 *     will be thrown.
 *    </li>
 *    <li>
 *      The final mechanism is to attempt to instantiate the <code>Class</code> specified by
 *      {@link #DATATYPEFACTORY_IMPLEMENTATION_CLASS}.
 *      Any Exception thrown during the instantiation process is wrapped as a {@link DatatypeConfigurationException}.
 *    </li>
 * </ol>
 *
 * @author <a href="mailto:Joseph.Fialli@Sun.COM">Joseph Fialli</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Neeraj.Bajaj@sun.com">Neeraj Bajaj</a>
 *
 * @version $Revision: 1.13 $, $Date: 2010/03/11 23:10:53 $
 * @since 1.5
 */
public abstract class DatatypeFactory {

    /**
     * <p>Default property name as defined in JSR 206: Java(TM) API for XML Processing (JAXP) 1.3.</p>
     *
     * <p>Default value is <code>javax.xml.datatype.DatatypeFactory</code>.</p>
     */
    public static final String DATATYPEFACTORY_PROPERTY =
            // We use a String constant here, rather than calling
            // DatatypeFactory.class.getName() - in order to make javadoc
            // generate a See Also: Constant Field Value link.
            "javax.xml.datatype.DatatypeFactory";

    /**
     * <p>Default implementation class name as defined in
     * <em>JSR 206: Java(TM) API for XML Processing (JAXP) 1.3</em>.</p>
     *
     * <p>Implementers should specify the name of an appropriate class
     * to be instantiated if no other implementation resolution mechanism
     * succeeds.</p>
     *
     * <p>Users should not refer to this field; it is intended only to
     * document a factory implementation detail.
     * </p>
     */
    public static final String DATATYPEFACTORY_IMPLEMENTATION_CLASS =
        // We use new String() here to prevent javadoc from generating
        // a See Also: Constant Field Value link.
        new String("com.sun.org.apache.xerces.internal.jaxp.datatype.DatatypeFactoryImpl");

    /**
     * http://www.w3.org/TR/xpath-datamodel/#xdtschema defines two regexps
     * to constrain the value space of dayTimeDuration ([^YM]*[DT].*)
     * and yearMonthDuration ([^DT]*). Note that these expressions rely on
```

```
     * the fact that the value must be an xs:Duration, they simply exclude
     * some Durations.
     */
    private static final Pattern XDTSCHEMA_YMD =
        Pattern.compile("[^DT]*");

    private static final Pattern XDTSCHEMA_DTD =
        Pattern.compile("[^YM]*[DT].*");

    /**
     * <p>Protected constructor to prevent instaniation outside of package.</p>
     *
     * <p>Use {@link #newInstance()} to create a <code>DatatypeFactory</code>.</p>
     */
    protected DatatypeFactory() {
    }

    /**
     * <p>Obtain a new instance of a <code>DatatypeFactory</code>.</p>
     *
     * <p>The implementation resolution mechanisms are <a href="#DatatypeFactory.newInstance">
       defined</a> in this
     * <code>Class</code>'s documentation.</p>
     *
     * @return New instance of a <code>DatatypeFactory</code>
     *
     * @throws DatatypeConfigurationException If the implementation is not
     *   available or cannot be instantiated.
     *
     * @see #newInstance(String factoryClassName, ClassLoader classLoader)
     */
    public static DatatypeFactory newInstance()
            throws DatatypeConfigurationException {

            return FactoryFinder.find(
                    /* The default property name according to the JAXP spec */
                    DatatypeFactory.class,
                    /* The fallback implementation class name */
                    DATATYPEFACTORY_IMPLEMENTATION_CLASS);
    }

    /**
     * <p>Obtain a new instance of a <code>DatatypeFactory</code> from class name.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
     * should be loaded.</p>
     *
     * <p>Once an application has obtained a reference to a <code>DatatypeFactory</code>
     * it can use the factory to configure and obtain datatype instances.</P>
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
```

```
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <code>javax.xml.datatype.DatatypeFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     *                    current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @return New instance of a <code>DatatypeFactory</code>
     *
     * @throws DatatypeConfigurationException if <code>factoryClassName</code> is <code>null</code>, or
     *                                  the factory class cannot be loaded, instantiated.
     *
     * @see #newInstance()
     *
     * @since 1.6
     */
    public static DatatypeFactory newInstance(String factoryClassName, ClassLoader classLoader)
        throws DatatypeConfigurationException {
        return FactoryFinder.newInstance(DatatypeFactory.class,
                    factoryClassName, classLoader, false);
     }

    /**
     * <p>Obtain a new instance of a <code>Duration</code>
     * specifying the <code>Duration</code> as its string representation, "PnYnMnDTnHnMnS",
     * as defined in XML Schema 1.0 section 3.2.6.1.</p>
     *
     * <p>XML Schema Part 2: Datatypes, 3.2.6 duration, defines <code>duration</code> as:</p>
     * <blockquote>
     * duration represents a duration of time.
     * The value space of duration is a six-dimensional space where the coordinates designate the
     * Gregorian year, month, day, hour, minute, and second components defined in Section 5.5.3.2 of [ISO 8601], respectively.
     * These components are ordered in their significance by their order of appearance i.e. as
     * year, month, day, hour, minute, and second.
     * </blockquote>
     * <p>All six values are set and available from the created {@link Duration}</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating implementation limits
```

```
     * if implementation capacities are exceeded.</p>
     *
     * @param lexicalRepresentation <code>String</code> representation of a <code>Duration</code>.
     *
     * @return New <code>Duration</code> created from parsing the <code>lexicalRepresentation</code>.
     *
     * @throws IllegalArgumentException If <code>lexicalRepresentation</code> is not a valid representation of a <code>Duration</code>.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     * @throws NullPointerException if <code>lexicalRepresentation</code> is <code>null</code>.
     */
    public abstract Duration newDuration(final String lexicalRepresentation);

    /**
     * <p>Obtain a new instance of a <code>Duration</code>
     * specifying the <code>Duration</code> as milliseconds.</p>
     *
     * <p>XML Schema Part 2: Datatypes, 3.2.6 duration, defines <code>duration</code> as:</p>
     * <blockquote>
     * duration represents a duration of time.
     * The value space of duration is a six-dimensional space where the coordinates designate the
     * Gregorian year, month, day, hour, minute, and second components defined in Section 5.5.3.2 of [ISO 8601], respectively.
     * These components are ordered in their significance by their order of appearance i.e. as
     * year, month, day, hour, minute, and second.
     * </blockquote>
     * <p>All six values are set by computing their values from the specified milliseconds
     * and are available using the <code>get</code> methods of  the created {@link Duration}.
     * The values conform to and are defined by:</p>
     * <ul>
     *   <li>ISO 8601:2000(E) Section 5.5.3.2 Alternative format</li>
     *   <li><a href="http://www.w3.org/TR/xmlschema-2/#isoformats">
     *     W3C XML Schema 1.0 Part 2, Appendix D, ISO 8601 Date and Time Formats</a>
     *   </li>
     *   <li>{@link XMLGregorianCalendar}  Date/Time Datatype Field Mapping Between XML Schema 1.0 and Java Representation</li>
     * </ul>
     *
     * <p>The default start instance is defined by {@link GregorianCalendar}'s use of the start of the epoch: i.e.,
     * {@link java.util.Calendar#YEAR} = 1970,
     * {@link java.util.Calendar#MONTH} = {@link java.util.Calendar#JANUARY},
     * {@link java.util.Calendar#DATE} = 1, etc.
     * This is important as there are variations in the Gregorian Calendar,
     * e.g. leap years have different days in the month = {@link java.util.Calendar#FEBRUARY}
     * so the result of {@link Duration#getMonths()} and {@link Duration#getDays()} can be influenced.</p>
     *
     * @param durationInMilliSeconds Duration in milliseconds to create.
     *
     * @return New <code>Duration</code> representing <code>durationInMilliSeconds</code>.
     */
```

```java
    public abstract Duration newDuration(final long durationInMilliSeconds);

    /**
     * <p>Obtain a new instance of a <code>Duration</code>
     * specifying the <code>Duration</code> as isPositive, years, months, days, hours, minutes,
         seconds.</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or
         small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating
         implementation limits
     * if implementation capacities are exceeded.</p>
     *
     * <p>A <code>null</code> value indicates that field is not set.</p>
     *
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param years of this <code>Duration</code>
     * @param months of this <code>Duration</code>
     * @param days of this <code>Duration</code>
     * @param hours of this <code>Duration</code>
     * @param minutes of this <code>Duration</code>
     * @param seconds of this <code>Duration</code>
     *
     * @return New <code>Duration</code> created from the specified values.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if all the fields (years, months, ...) are null or
     * if any of the fields is negative.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     */
    public abstract Duration newDuration(
            final boolean isPositive,
            final BigInteger years,
            final BigInteger months,
            final BigInteger days,
            final BigInteger hours,
            final BigInteger minutes,
            final BigDecimal seconds);

    /**
     * <p>Obtain a new instance of a <code>Duration</code>
     * specifying the <code>Duration</code> as isPositive, years, months, days, hours, minutes,
         seconds.</p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param years of this <code>Duration</code>
     * @param months of this <code>Duration</code>
     * @param days of this <code>Duration</code>
```

```
     * @param hours of this <code>Duration</code>
     * @param minutes of this <code>Duration</code>
     * @param seconds of this <code>Duration</code>
     *
     * @return New <code>Duration</code> created from the specified values.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if any of the fields is negative.
     *
     * @see #newDuration(
     *   boolean isPositive,
     *   BigInteger years,
     *   BigInteger months,
     *   BigInteger days,
     *   BigInteger hours,
     *   BigInteger minutes,
     *   BigDecimal seconds)
     */
    public Duration newDuration(
            final boolean isPositive,
            final int years,
            final int months,
            final int days,
            final int hours,
            final int minutes,
            final int seconds) {

            // years may not be set
            BigInteger realYears = (years != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.
                valueOf((long) years) : null;

            // months may not be set
            BigInteger realMonths = (months != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.
                valueOf((long) months) : null;

            // days may not be set
            BigInteger realDays = (days != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.valueOf
                ((long) days) : null;

            // hours may not be set
            BigInteger realHours = (hours != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.
                valueOf((long) hours) : null;

            // minutes may not be set
            BigInteger realMinutes = (minutes != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.
                valueOf((long) minutes) : null;

            // seconds may not be set
            BigDecimal realSeconds = (seconds != DatatypeConstants.FIELD_UNDEFINED) ? BigDecimal.
                valueOf((long) seconds) : null;

                    return newDuration(
                            isPositive,
```

```
                            realYears,
                            realMonths,
                            realDays,
                            realHours,
                            realMinutes,
                            realSeconds
                    );
            }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:dayTimeDuration</code> by parsing its <code>String</code> representation,
     * "<em>PnDTnHnMnS</em>", <a href="http://www.w3.org/TR/xpath-datamodel#dayTimeDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:dayTimeDuration</a>.</p>
     *
     * <p>The datatype <code>xdt:dayTimeDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only day, hour, minute, and second components.
     * This datatype resides in the namespace <code>http://www.w3.org/2003/11/xpath-datatypes</code>.</p>
     *
     * <p>All four values are set and available from the created {@link Duration}</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating implementation limits
     * if implementation capacities are exceeded.</p>
     *
     * @param lexicalRepresentation Lexical representation of a duration.
     *
     * @return New <code>Duration</code> created using the specified <code>lexicalRepresentation</code>.
     *
     * @throws IllegalArgumentException If <code>lexicalRepresentation</code> is not a valid representation of a <code>Duration</code> expressed only in terms of days and time.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     * @throws NullPointerException If <code>lexicalRepresentation</code> is <code>null</code>.
     */
    public Duration newDurationDayTime(final String lexicalRepresentation) {
        // lexicalRepresentation must be non-null
        if (lexicalRepresentation == null) {
            throw new NullPointerException(
                "Trying to create an xdt:dayTimeDuration with an invalid"
                + " lexical representation of \"null\"");
        }

        // test lexicalRepresentation against spec regex
        Matcher matcher = XDTSCHEMA_DTD.matcher(lexicalRepresentation);
        if (!matcher.matches()) {
            throw new IllegalArgumentException(
                "Trying to create an xdt:dayTimeDuration with an invalid"
                + " lexical representation of \"" + lexicalRepresentation
```

```
                + "\", data model requires years and months only.");
        }

        return newDuration(lexicalRepresentation);
    }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:dayTimeDuration</code> using the specified milliseconds as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#dayTimeDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:dayTimeDuration</a>.</p>
     *
     * <p>The datatype <code>xdt:dayTimeDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only day, hour, minute, and second components.
     * This datatype resides in the namespace <code>http://www.w3.org/2003/11/xpath-datatypes</code>.</p>
     *
     * <p>All four values are set by computing their values from the specified milliseconds
     * and are available using the <code>get</code> methods of  the created {@link Duration}.
     * The values conform to and are defined by:</p>
     * <ul>
     *   <li>ISO 8601:2000(E) Section 5.5.3.2 Alternative format</li>
     *   <li><a href="http://www.w3.org/TR/xmlschema-2/#isoformats">
     *     W3C XML Schema 1.0 Part 2, Appendix D, ISO 8601 Date and Time Formats</a>
     *   </li>
     *   <li>{@link XMLGregorianCalendar}  Date/Time Datatype Field Mapping Between XML Schema 1.0 and Java Representation</li>
     * </ul>
     *
     * <p>The default start instance is defined by {@link GregorianCalendar}'s use of the start of the epoch: i.e.,
     * {@link java.util.Calendar#YEAR} = 1970,
     * {@link java.util.Calendar#MONTH} = {@link java.util.Calendar#JANUARY},
     * {@link java.util.Calendar#DATE} = 1, etc.
     * This is important as there are variations in the Gregorian Calendar,
     * e.g. leap years have different days in the month = {@link java.util.Calendar#FEBRUARY}
     * so the result of {@link Duration#getDays()} can be influenced.</p>
     *
     * <p>Any remaining milliseconds after determining the day, hour, minute and second are discarded.</p>
     *
     * @param durationInMilliseconds Milliseconds of <code>Duration</code> to create.
     *
     * @return New <code>Duration</code> created with the specified <code>durationInMilliseconds</code>.
     *
     * @see <a href="http://www.w3.org/TR/xpath-datamodel#dayTimeDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:dayTimeDuration</a>
     */
    public Duration newDurationDayTime(final long durationInMilliseconds) {

            return newDuration(durationInMilliseconds);
    }
```

```java

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:dayTimeDuration</code> using the specified
     * <code>day</code>, <code>hour</code>, <code>minute</code> and <code>second</code> as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#dayTimeDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:dayTimeDuration</a>.</p>
     *
     * <p>The datatype <code>xdt:dayTimeDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only day, hour, minute, and second components.
     * This datatype resides in the namespace <code>http://www.w3.org/2003/11/xpath-datatypes</code>.</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating implementation limits
     * if implementation capacities are exceeded.</p>
     *
     * <p>A <code>null</code> value indicates that field is not set.</p>
     *
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param day Day of <code>Duration</code>.
     * @param hour Hour of <code>Duration</code>.
     * @param minute Minute of <code>Duration</code>.
     * @param second Second of <code>Duration</code>.
     *
     * @return New <code>Duration</code> created with the specified <code>day</code>, <code>hour</code>, <code>minute</code>
     * and <code>second</code>.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if all the fields (day, hour, ...) are null or
     * if any of the fields is negative.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     */
    public Duration newDurationDayTime(
            final boolean isPositive,
            final BigInteger day,
            final BigInteger hour,
            final BigInteger minute,
            final BigInteger second) {

            return newDuration(
                    isPositive,
                    null,  // years
                    null, // months
                    day,
                    hour,
                    minute,
```

```
                    (second != null)? new BigDecimal(second):null
            );
    }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:dayTimeDuration</code> using the specified
     * <code>day</code>, <code>hour</code>, <code>minute</code> and <code>second</code> as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#dayTimeDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:dayTimeDuration</a>.</p>
     *
     * <p>The datatype <code>xdt:dayTimeDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only day, hour, minute, and second components.
     * This datatype resides in the namespace <code>http://www.w3.org/2003/11/xpath-datatypes</code>.</p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param day Day of <code>Duration</code>.
     * @param hour Hour of <code>Duration</code>.
     * @param minute Minute of <code>Duration</code>.
     * @param second Second of <code>Duration</code>.
     *
     * @return New <code>Duration</code> created with the specified <code>day</code>, <code>hour</code>, <code>minute</code>
     * and <code>second</code>.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if any of the fields (day, hour, ...) is negative.
     */
    public Duration newDurationDayTime(
            final boolean isPositive,
            final int day,
            final int hour,
            final int minute,
            final int second) {

                    return newDurationDayTime(
                            isPositive,
                            BigInteger.valueOf((long) day),
                            BigInteger.valueOf((long) hour),
                            BigInteger.valueOf((long) minute),
                            BigInteger.valueOf((long) second)
                            );
            }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:yearMonthDuration</code> by parsing its <code>String</code> representation,
     * "<em>PnYnM</em>", <a href="http://www.w3.org/TR/xpath-datamodel#yearMonthDuration">
```

```
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:yearMonthDuration</a>.</p>
     *
     * <p>The datatype <code>xdt:yearMonthDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only year and month components.
     * This datatype resides in the namespace {@link javax.xml.XMLConstants#
         W3C_XPATH_DATATYPE_NS_URI}.</p>
     *
     * <p>Both values are set and available from the created {@link Duration}</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or
         small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating
         implementation limits
     * if implementation capacities are exceeded.</p>
     *
     * @param lexicalRepresentation Lexical representation of a duration.
     *
     * @return New <code>Duration</code> created using the specified <code>lexicalRepresentation</
         code>.
     *
     * @throws IllegalArgumentException If <code>lexicalRepresentation</code> is not a valid
         representation of a <code>Duration</code> expressed only in terms of years and months.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     * @throws NullPointerException If <code>lexicalRepresentation</code> is <code>null</code>.
     */
    public Duration newDurationYearMonth(
            final String lexicalRepresentation) {

        // lexicalRepresentation must be non-null
        if (lexicalRepresentation == null) {
            throw new NullPointerException(
                    "Trying to create an xdt:yearMonthDuration with an invalid"
                    + " lexical representation of \"null\"");
        }

        // test lexicalRepresentation against spec regex
        Matcher matcher = XDTSCHEMA_YMD.matcher(lexicalRepresentation);
        if (!matcher.matches()) {
            throw new IllegalArgumentException(
                    "Trying to create an xdt:yearMonthDuration with an invalid"
                    + " lexical representation of \"" + lexicalRepresentation
                    + "\", data model requires days and times only.");
        }

        return newDuration(lexicalRepresentation);
    }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:yearMonthDuration</code> using the
         specified milliseconds as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#yearMonthDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:yearMonthDuration</a>.</p>
```

```
     *
     * <p>The datatype <code>xdt:yearMonthDuration</code> is a subtype of <code>xs:duration</code>
     * whose lexical representation contains only year and month components.
     * This datatype resides in the namespace {@link javax.xml.XMLConstants#
         W3C_XPATH_DATATYPE_NS_URI}.</p>
     *
     * <p>Both values are set by computing their values from the specified milliseconds
     * and are available using the <code>get</code> methods of  the created {@link Duration}.
     * The values conform to and are defined by:</p>
     * <ul>
     *   <li>ISO 8601:2000(E) Section 5.5.3.2 Alternative format</li>
     *   <li><a href="http://www.w3.org/TR/xmlschema-2/#isoformats">
     *     W3C XML Schema 1.0 Part 2, Appendix D, ISO 8601 Date and Time Formats</a>
     *   </li>
     *   <li>{@link XMLGregorianCalendar}  Date/Time Datatype Field Mapping Between XML Schema 1.0
         and Java Representation</li>
     * </ul>
     *
     * <p>The default start instance is defined by {@link GregorianCalendar}'s use of the start of
         the epoch: i.e.,
     * {@link java.util.Calendar#YEAR} = 1970,
     * {@link java.util.Calendar#MONTH} = {@link java.util.Calendar#JANUARY},
     * {@link java.util.Calendar#DATE} = 1, etc.
     * This is important as there are variations in the Gregorian Calendar,
     * e.g. leap years have different days in the month = {@link java.util.Calendar#FEBRUARY}
     * so the result of {@link Duration#getMonths()} can be influenced.</p>
     *
     * <p>Any remaining milliseconds after determining the year and month are discarded.</p>
     *
     * @param durationInMilliseconds Milliseconds of <code>Duration</code> to create.
     *
     * @return New <code>Duration</code> created using the specified <code>durationInMilliseconds</
         code>.
     */
    public Duration newDurationYearMonth(
            final long durationInMilliseconds) {

        // create a Duration that only has sign, year & month
        // Duration is immutable, so need to create a new Duration
        // implementations may override this method in a more efficient way
        Duration fullDuration = newDuration(durationInMilliseconds);
        boolean isPositive = (fullDuration.getSign() == -1) ? false : true;
        BigInteger years =
            (BigInteger) fullDuration.getField(DatatypeConstants.YEARS);
        if (years == null) { years = BigInteger.ZERO; }
        BigInteger months =
            (BigInteger) fullDuration.getField(DatatypeConstants.MONTHS);
        if (months == null) { months = BigInteger.ZERO; }

        return newDurationYearMonth(isPositive, years, months);
    }

    /**
```

```
     * <p>Create a <code>Duration</code> of type <code>xdt:yearMonthDuration</code> using the specified
     * <code>year</code> and <code>month</code> as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#yearMonthDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:yearMonthDuration</a>.</p>
     *
     * <p>The XML Schema specification states that values can be of an arbitrary size.
     * Implementations may chose not to or be incapable of supporting arbitrarily large and/or small values.
     * An {@link UnsupportedOperationException} will be thrown with a message indicating implementation limits
     * if implementation capacities are exceeded.</p>
     *
     * <p>A <code>null</code> value indicates that field is not set.</p>
     *
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param year Year of <code>Duration</code>.
     * @param month Month of <code>Duration</code>.
     *
     * @return New <code>Duration</code> created using the specified <code>year</code> and <code>month</code>.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if all of the fields (year, month) are null or
     * if any of the fields is negative.
     * @throws UnsupportedOperationException If implementation cannot support requested values.
     */
    public Duration newDurationYearMonth(
            final boolean isPositive,
            final BigInteger year,
            final BigInteger month) {

            return newDuration(
                    isPositive,
                    year,
                    month,
                    null, // days
                    null, // hours
                    null, // minutes
                    null  // seconds
            );
    }

    /**
     * <p>Create a <code>Duration</code> of type <code>xdt:yearMonthDuration</code> using the specified
     * <code>year</code> and <code>month</code> as defined in
     * <a href="http://www.w3.org/TR/xpath-datamodel#yearMonthDuration">
     *   XQuery 1.0 and XPath 2.0 Data Model, xdt:yearMonthDuration</a>.</p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
```

```
     * @param isPositive Set to <code>false</code> to create a negative duration. When the length
     *   of the duration is zero, this parameter will be ignored.
     * @param year Year of <code>Duration</code>.
     * @param month Month of <code>Duration</code>.
     *
     * @return New <code>Duration</code> created using the specified <code>year</code> and <code>month</code>.
     *
     * @throws IllegalArgumentException If the values are not a valid representation of a
     * <code>Duration</code>: if any of the fields (year, month) is negative.
     */
    public Duration newDurationYearMonth(
            final boolean isPositive,
            final int year,
            final int month) {

            return newDurationYearMonth(
                    isPositive,
                    BigInteger.valueOf((long) year),
                    BigInteger.valueOf((long) month));
            }

    /**
     * <p>Create a new instance of an <code>XMLGregorianCalendar</code>.</p>
     *
     * <p>All date/time datatype fields set to {@link DatatypeConstants#FIELD_UNDEFINED} or null.</p>
     *
     * @return New <code>XMLGregorianCalendar</code> with all date/time datatype fields set to
     *   {@link DatatypeConstants#FIELD_UNDEFINED} or null.
     */
    public abstract XMLGregorianCalendar newXMLGregorianCalendar();

    /**
     * <p>Create a new XMLGregorianCalendar by parsing the String as a lexical representation.</p>
     *
     * <p>Parsing the lexical string representation is defined in
     * <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">XML Schema 1.0 Part 2, Section 3.2.[7-14].1,
     * <em>Lexical Representation</em>.</a></p>
     *
     * <p>The string representation may not have any leading and trailing whitespaces.</p>
     *
     * <p>The parsing is done field by field so that
     * the following holds for any lexically correct String x:</p>
     * <pre>
     * newXMLGregorianCalendar(x).toXMLFormat().equals(x)
     * </pre>
     * <p>Except for the noted lexical/canonical representation mismatches
     * listed in <a href="http://www.w3.org/2001/05/xmlschema-errata#e2-45">
     * XML Schema 1.0 errata, Section 3.2.7.2</a>.</p>
     *
     * @param lexicalRepresentation Lexical representation of one the eight XML Schema date/time
```

```
        datatypes.
     *
     * @return <code>XMLGregorianCalendar</code> created from the <code>lexicalRepresentation</code>.
     *
     * @throws IllegalArgumentException If the <code>lexicalRepresentation</code> is not a valid <code>XMLGregorianCalendar</code>.
     * @throws NullPointerException If <code>lexicalRepresentation</code> is <code>null</code>.
     */
    public abstract XMLGregorianCalendar newXMLGregorianCalendar(final String lexicalRepresentation);

    /**
     * <p>Create an <code>XMLGregorianCalendar</code> from a {@link GregorianCalendar}.</p>
     *
     * <table border="2" rules="all" cellpadding="2">
     *   <thead>
     *     <tr>
     *       <th align="center" colspan="2">
     *          Field by Field Conversion from
     *          {@link GregorianCalendar} to an {@link XMLGregorianCalendar}
     *       </th>
     *     </tr>
     *     <tr>
     *        <th><code>java.util.GregorianCalendar</code> field</th>
     *        <th><code>javax.xml.datatype.XMLGregorianCalendar</code> field</th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td><code>ERA == GregorianCalendar.BC ? -YEAR : YEAR</code></td>
     *       <td>{@link XMLGregorianCalendar#setYear(int year)}</td>
     *     </tr>
     *     <tr>
     *       <td><code>MONTH + 1</code></td>
     *       <td>{@link XMLGregorianCalendar#setMonth(int month)}</td>
     *     </tr>
     *     <tr>
     *       <td><code>DAY_OF_MONTH</code></td>
     *       <td>{@link XMLGregorianCalendar#setDay(int day)}</td>
     *     </tr>
     *     <tr>
     *       <td><code>HOUR_OF_DAY, MINUTE, SECOND, MILLISECOND</code></td>
     *       <td>{@link XMLGregorianCalendar#setTime(int hour, int minute, int second, BigDecimal fractional)}</td>
     *     </tr>
     *     <tr>
     *       <td>
     *         <code>(ZONE_OFFSET + DST_OFFSET) / (60*1000)</code><br/>
     *         <em>(in minutes)</em>
     *       </td>
     *       <td>{@link XMLGregorianCalendar#setTimezone(int offset)}<sup><em>*</em></sup>
     *       </td>
```

```
     *     </tr>
     *   </tbody>
     * </table>
     * <p><em>*</em>conversion loss of information. It is not possible to represent
     * a <code>java.util.GregorianCalendar</code> daylight savings timezone id in the
     * XML Schema 1.0 date/time datatype representation.</p>
     *
     * <p>To compute the return value's <code>TimeZone</code> field,
     * <ul>
     * <li>when <code>this.getTimezone() != FIELD_UNDEFINED</code>,
     * create a <code>java.util.TimeZone</code> with a custom timezone id
     * using the <code>this.getTimezone()</code>.</li>
     * <li>else use the <code>GregorianCalendar</code> default timezone value
     * for the host is defined as specified by
     * <code>java.util.TimeZone.getDefault()</code>.</li></p>
     *
     * @param cal <code>java.util.GregorianCalendar</code> used to create <code>
         XMLGregorianCalendar</code>
     *
     * @return <code>XMLGregorianCalendar</code> created from <code>java.util.GregorianCalendar</
         code>
     *
     * @throws NullPointerException If <code>cal</code> is <code>null</code>.
     */
    public abstract XMLGregorianCalendar newXMLGregorianCalendar(final GregorianCalendar cal);

    /**
     * <p>Constructor allowing for complete value spaces allowed by
     * W3C XML Schema 1.0 recommendation for xsd:dateTime and related
     * builtin datatypes. Note that <code>year</code> parameter supports
     * arbitrarily large numbers and fractionalSecond has infinite
     * precision.</p>
     *
     * <p>A <code>null</code> value indicates that field is not set.</p>
     *
     * @param year of <code>XMLGregorianCalendar</code> to be created.
     * @param month of <code>XMLGregorianCalendar</code> to be created.
     * @param day of <code>XMLGregorianCalendar</code> to be created.
     * @param hour of <code>XMLGregorianCalendar</code> to be created.
     * @param minute of <code>XMLGregorianCalendar</code> to be created.
     * @param second of <code>XMLGregorianCalendar</code> to be created.
     * @param fractionalSecond of <code>XMLGregorianCalendar</code> to be created.
     * @param timezone of <code>XMLGregorianCalendar</code> to be created.
     *
     * @return <code>XMLGregorianCalendar</code> created from specified values.
     *
     * @throws IllegalArgumentException If any individual parameter's value is outside the maximum
         value constraint for the field
     *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
     *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code>
         instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     */
```

```java
    public abstract XMLGregorianCalendar newXMLGregorianCalendar(
            final BigInteger year,
            final int month,
            final int day,
            final int hour,
            final int minute,
            final int second,
            final BigDecimal fractionalSecond,
            final int timezone);

    /**
     * <p>Constructor of value spaces that a
     * <code>java.util.GregorianCalendar</code> instance would need to convert to an
     * <code>XMLGregorianCalendar</code> instance.</p>
     *
     * <p><code>XMLGregorianCalendar eon</code> and
     * <code>fractionalSecond</code> are set to <code>null</code></p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
     * @param year of <code>XMLGregorianCalendar</code> to be created.
     * @param month of <code>XMLGregorianCalendar</code> to be created.
     * @param day of <code>XMLGregorianCalendar</code> to be created.
     * @param hour of <code>XMLGregorianCalendar</code> to be created.
     * @param minute of <code>XMLGregorianCalendar</code> to be created.
     * @param second of <code>XMLGregorianCalendar</code> to be created.
     * @param millisecond of <code>XMLGregorianCalendar</code> to be created.
     * @param timezone of <code>XMLGregorianCalendar</code> to be created.
     *
     * @return <code>XMLGregorianCalendar</code> created from specified values.
     *
     * @throws IllegalArgumentException If any individual parameter's value is outside the maximum value constraint for the field
     *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
     *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code> instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     */
    public XMLGregorianCalendar newXMLGregorianCalendar(
            final int year,
            final int month,
            final int day,
            final int hour,
            final int minute,
            final int second,
            final int millisecond,
            final int timezone) {

            // year may be undefined
            BigInteger realYear = (year != DatatypeConstants.FIELD_UNDEFINED) ? BigInteger.valueOf((long) year) : null;

            // millisecond may be undefined
```

```
        // millisecond must be >= 0 millisecond <= 1000
        BigDecimal realMillisecond = null; // undefined value
        if (millisecond != DatatypeConstants.FIELD_UNDEFINED) {
                if (millisecond < 0 || millisecond > 1000) {
                        throw new IllegalArgumentException(
                                                "javax.xml.datatype.DatatypeFactory#newXMLGregorianCalendar("
                                                + "int year, int month, int day, int hour, int minute, int second, int millisecond, int timezone)"
                                                + "with invalid millisecond: " + millisecond
                                                );
                }

                realMillisecond = BigDecimal.valueOf((long) millisecond).movePointLeft(3);
        }

        return newXMLGregorianCalendar(
                realYear,
                month,
                day,
                hour,
                minute,
                second,
                realMillisecond,
                timezone
        );
}

/**
 * <p>Create a Java representation of XML Schema builtin datatype <code>date</code> or <code>g*</code>.</p>
 *
 * <p>For example, an instance of <code>gYear</code> can be created invoking this factory
 * with <code>month</code> and <code>day</code> parameters set to
 * {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
 *
 * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
 *
 * @param year of <code>XMLGregorianCalendar</code> to be created.
 * @param month of <code>XMLGregorianCalendar</code> to be created.
 * @param day of <code>XMLGregorianCalendar</code> to be created.
 * @param timezone offset in minutes. {@link DatatypeConstants#FIELD_UNDEFINED} indicates optional field is not set.
 *
 * @return <code>XMLGregorianCalendar</code> created from parameter values.
 *
 * @see DatatypeConstants#FIELD_UNDEFINED
 *
 * @throws IllegalArgumentException If any individual parameter's value is outside the maximum value constraint for the field
 *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
 *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code>
```

```
        instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     */
    public XMLGregorianCalendar newXMLGregorianCalendarDate(
            final int year,
            final int month,
            final int day,
            final int timezone) {

            return newXMLGregorianCalendar(
                    year,
                    month,
                    day,
                    DatatypeConstants.FIELD_UNDEFINED, // hour
                    DatatypeConstants.FIELD_UNDEFINED, // minute
                    DatatypeConstants.FIELD_UNDEFINED, // second
                    DatatypeConstants.FIELD_UNDEFINED, // millisecond
                    timezone);
            }

    /**
     * <p>Create a Java instance of XML Schema builtin datatype <code>time</code>.</p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
     * @param hours number of hours
     * @param minutes number of minutes
     * @param seconds number of seconds
     * @param timezone offset in minutes. {@link DatatypeConstants#FIELD_UNDEFINED} indicates
        optional field is not set.
     *
     * @return <code>XMLGregorianCalendar</code> created from parameter values.
     *
     * @throws IllegalArgumentException If any individual parameter's value is outside the maximum
        value constraint for the field
     *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
     *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code>
        instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     *
     * @see DatatypeConstants#FIELD_UNDEFINED
     */
    public XMLGregorianCalendar newXMLGregorianCalendarTime(
            final int hours,
            final int minutes,
            final int seconds,
            final int timezone) {

            return newXMLGregorianCalendar(
                    DatatypeConstants.FIELD_UNDEFINED, // Year
                    DatatypeConstants.FIELD_UNDEFINED, // Month
                    DatatypeConstants.FIELD_UNDEFINED, // Day
                    hours,
```

```
                    minutes,
                    seconds,
                    DatatypeConstants.FIELD_UNDEFINED, //Millisecond
                    timezone);
    }

    /**
     * <p>Create a Java instance of XML Schema builtin datatype time.</p>
     *
     * <p>A <code>null</code> value indicates that field is not set.</p>
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
     *
     * @param hours number of hours
     * @param minutes number of minutes
     * @param seconds number of seconds
     * @param fractionalSecond value of <code>null</code> indicates that this optional field is not set.
     * @param timezone offset in minutes. {@link DatatypeConstants#FIELD_UNDEFINED} indicates optional field is not set.
     *
     * @return <code>XMLGregorianCalendar</code> created from parameter values.
     *
     * @see DatatypeConstants#FIELD_UNDEFINED
     *
     * @throws IllegalArgumentException If any individual parameter's value is outside the maximum value constraint for the field
     *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
     *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code> instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     */
    public XMLGregorianCalendar newXMLGregorianCalendarTime(
            final int hours,
            final int minutes,
            final int seconds,
            final BigDecimal fractionalSecond,
            final int timezone) {

            return newXMLGregorianCalendar(
                    null, // year
                    DatatypeConstants.FIELD_UNDEFINED, // month
                    DatatypeConstants.FIELD_UNDEFINED, // day
                    hours,
                    minutes,
                    seconds,
                    fractionalSecond,
                    timezone);
            }

    /**
     * <p>Create a Java instance of XML Schema builtin datatype time.</p>
     *
     * <p>A {@link DatatypeConstants#FIELD_UNDEFINED} value indicates that field is not set.</p>
```

```
     *
     * @param hours number of hours
     * @param minutes number of minutes
     * @param seconds number of seconds
     * @param milliseconds number of milliseconds
     * @param timezone offset in minutes. {@link DatatypeConstants#FIELD_UNDEFINED} indicates optional field is not set.
     *
     * @return <code>XMLGregorianCalendar</code> created from parameter values.
     *
     * @see DatatypeConstants#FIELD_UNDEFINED
     *
     * @throws IllegalArgumentException If any individual parameter's value is outside the maximum value constraint for the field
     *   as determined by the Date/Time Data Mapping table in {@link XMLGregorianCalendar}
     *   or if the composite values constitute an invalid <code>XMLGregorianCalendar</code> instance
     *   as determined by {@link XMLGregorianCalendar#isValid()}.
     */
    public XMLGregorianCalendar newXMLGregorianCalendarTime(
            final int hours,
            final int minutes,
            final int seconds,
            final int milliseconds,
            final int timezone) {

            // millisecond may be undefined
            // millisecond must be >= 0 millisecond <= 1000
            BigDecimal realMilliseconds = null; // undefined value
            if (milliseconds != DatatypeConstants.FIELD_UNDEFINED) {
                    if (milliseconds < 0 || milliseconds > 1000) {
                            throw new IllegalArgumentException(
                                                    "javax.xml.datatype.DatatypeFactory#newXMLGregorianCalendarTime("
                                                    + "int hours, int minutes, int seconds, int milliseconds, int timezone)"
                                                    + "with invalid milliseconds: " + milliseconds
                                                    );
                    }

                    realMilliseconds = BigDecimal.valueOf((long) milliseconds).movePointLeft(3);
            }

            return newXMLGregorianCalendarTime(
                    hours,
                    minutes,
                    seconds,
                    realMilliseconds,
                    timezone
            );
    }
}
```

14.4 Duration.java

Secuencia 148: javax/xml/datatype/Duration.java

```
/*
 * Copyright (c) 2003, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

import javax.xml.namespace.QName;

/**
 * <p>Immutable representation of a time span as defined in
 * the W3C XML Schema 1.0 specification.</p>
 *
 * <p>A Duration object represents a period of Gregorian time,
 * which consists of six fields (years, months, days, hours,
 * minutes, and seconds) plus a sign (+/-) field.</p>
 *
 * <p>The first five fields have non-negative (>=0) integers or null
 * (which represents that the field is not set),
 * and the seconds field has a non-negative decimal or null.
 * A negative sign indicates a negative duration.</p>
 *
 * <p>This class provides a number of methods that make it easy
 * to use for the duration datatype of XML Schema 1.0 with
```

```
 * the errata.</p>
 *
 * <h2>Order relationship</h2>
 * <p>Duration objects only have partial order, where two values A and B
 * maybe either:</p>
 * <ol>
 *  <li>A&lt;B (A is shorter than B)
 *  <li>A&gt;B (A is longer than B)
 *  <li>A==B   (A and B are of the same duration)
 *  <li>A&lt;>B (Comparison between A and B is indeterminate)
 * </ol>
 *
 * <p>For example, 30 days cannot be meaningfully compared to one month.
 * The {@link #compare(Duration duration)} method implements this
 * relationship.</p>
 *
 * <p>See the {@link #isLongerThan(Duration)} method for details about
 * the order relationship among <code>Duration</code> objects.</p>
 *
 * <h2>Operations over Duration</h2>
 * <p>This class provides a set of basic arithmetic operations, such
 * as addition, subtraction and multiplication.
 * Because durations don't have total order, an operation could
 * fail for some combinations of operations. For example, you cannot
 * subtract 15 days from 1 month. See the javadoc of those methods
 * for detailed conditions where this could happen.</p>
 *
 * <p>Also, division of a duration by a number is not provided because
 * the <code>Duration</code> class can only deal with finite precision
 * decimal numbers. For example, one cannot represent 1 sec divided by 3.</p>
 *
 * <p>However, you could substitute a division by 3 with multiplying
 * by numbers such as 0.3 or 0.333.</p>
 *
 * <h2>Range of allowed values</h2>
 * <p>
 * Because some operations of <code>Duration</code> rely on {@link Calendar}
 * even though {@link Duration} can hold very large or very small values,
 * some of the methods may not work correctly on such <code>Duration</code>s.
 * The impacted methods document their dependency on {@link Calendar}.
 *
 * @author <a href="mailto:Joseph.Fialli@Sun.COM">Joseph Fialli</a>
 * @author <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Sunitha.Reddy@Sun.com">Sunitha Reddy</a>
 * @see XMLGregorianCalendar#add(Duration)
 * @since 1.5
 */
public abstract class Duration {

    /**
     * <p>Debugging <code>true</code> or <code>false</code>.</p>
     */
```

```
    private static final boolean DEBUG = true;

    /**
     * Default no-arg constructor.
     *
     * <p>Note: Always use the {@link DatatypeFactory} to
     * construct an instance of <code>Duration</code>.
     * The constructor on this class cannot be guaranteed to
     * produce an object with a consistent state and may be
     * removed in the future.</p>
     */
    public Duration() {
    }

    /**
     * <p>Return the name of the XML Schema date/time type that this instance
     * maps to. Type is computed based on fields that are set,
     * i.e. {@link #isSet(DatatypeConstants.Field field)} == <code>true</code>.</p>
     *
     * <table border="2" rules="all" cellpadding="2">
     *   <thead>
     *     <tr>
     *       <th align="center" colspan="7">
     *         Required fields for XML Schema 1.0 Date/Time Datatypes.<br/>
     *         <i>(timezone is optional for all date/time datatypes)</i>
     *       </th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td>Datatype</td>
     *       <td>year</td>
     *       <td>month</td>
     *       <td>day</td>
     *       <td>hour</td>
     *       <td>minute</td>
     *       <td>second</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#DURATION}</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#DURATION_DAYTIME}</td>
     *       <td></td>
     *       <td></td>
     *       <td>X</td>
     *       <td>X</td>
```

```
     *       <td>X</td>
     *       <td>X</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#DURATION_YEARMONTH}</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *   </tbody>
     * </table>
     *
     * @return one of the following constants:
     *   {@link DatatypeConstants#DURATION},
     *   {@link DatatypeConstants#DURATION_DAYTIME} or
     *   {@link DatatypeConstants#DURATION_YEARMONTH}.
     *
     * @throws IllegalStateException If the combination of set fields does not match one of the XML
          Schema date/time datatypes.
     */
    public QName getXMLSchemaType() {

        boolean yearSet = isSet(DatatypeConstants.YEARS);
        boolean monthSet = isSet(DatatypeConstants.MONTHS);
        boolean daySet = isSet(DatatypeConstants.DAYS);
        boolean hourSet = isSet(DatatypeConstants.HOURS);
        boolean minuteSet = isSet(DatatypeConstants.MINUTES);
        boolean secondSet = isSet(DatatypeConstants.SECONDS);

        // DURATION
        if (yearSet
            && monthSet
            && daySet
            && hourSet
            && minuteSet
            && secondSet) {
            return DatatypeConstants.DURATION;
        }

        // DURATION_DAYTIME
        if (!yearSet
            && !monthSet
            && daySet
            && hourSet
            && minuteSet
            && secondSet) {
            return DatatypeConstants.DURATION_DAYTIME;
        }

        // DURATION_YEARMONTH
```

```
        if (yearSet
            && monthSet
            && !daySet
            && !hourSet
            && !minuteSet
            && !secondSet) {
            return DatatypeConstants.DURATION_YEARMONTH;
        }

        // nothing matches
        throw new IllegalStateException(
                "javax.xml.datatype.Duration#getXMLSchemaType():"
                + " this Duration does not match one of the XML Schema date/time datatypes:"
                + " year set = " + yearSet
                + " month set = " + monthSet
                + " day set = " + daySet
                + " hour set = " + hourSet
                + " minute set = " + minuteSet
                + " second set = " + secondSet
        );
    }

    /**
     * Returns the sign of this duration in -1,0, or 1.
     *
     * @return
     *      -1 if this duration is negative, 0 if the duration is zero,
     *      and 1 if the duration is positive.
     */
    public abstract int getSign();

    /**
     * <p>Get the years value of this <code>Duration</code> as an <code>int</code> or <code>0</code
       > if not present.</p>
     *
     * <p><code>getYears()</code> is a convenience method for
     * {@link #getField(DatatypeConstants.Field field) getField(DatatypeConstants.YEARS)}.</p>
     *
     * <p>As the return value is an <code>int</code>, an incorrect value will be returned for <code
       >Duration</code>s
     * with years that go beyond the range of an <code>int</code>.
     * Use {@link #getField(DatatypeConstants.Field field) getField(DatatypeConstants.YEARS)} to
       avoid possible loss of precision.</p>
     *
     * @return If the years field is present, return its value as an <code>int</code>, else return
       <code>0</code>.
     */
    public int getYears() {
        return getField(DatatypeConstants.YEARS).intValue();
    }

    /**
     * Obtains the value of the MONTHS field as an integer value,
```

```
     * or 0 if not present.
     *
     * This method works just like {@link #getYears()} except
     * that this method works on the MONTHS field.
     *
     * @return Months of this <code>Duration</code>.
     */
    public int getMonths() {
        return getField(DatatypeConstants.MONTHS).intValue();
    }

    /**
     * Obtains the value of the DAYS field as an integer value,
     * or 0 if not present.
     *
     * This method works just like {@link #getYears()} except
     * that this method works on the DAYS field.
     *
     * @return Days of this <code>Duration</code>.
     */
    public int getDays() {
        return getField(DatatypeConstants.DAYS).intValue();
    }

    /**
     * Obtains the value of the HOURS field as an integer value,
     * or 0 if not present.
     *
     * This method works just like {@link #getYears()} except
     * that this method works on the HOURS field.
     *
     * @return Hours of this <code>Duration</code>.
     *
     */
    public int getHours() {
        return getField(DatatypeConstants.HOURS).intValue();
    }

    /**
     * Obtains the value of the MINUTES field as an integer value,
     * or 0 if not present.
     *
     * This method works just like {@link #getYears()} except
     * that this method works on the MINUTES field.
     *
     * @return Minutes of this <code>Duration</code>.
     *
     */
    public int getMinutes() {
        return getField(DatatypeConstants.MINUTES).intValue();
    }

    /**
```

```
     * Obtains the value of the SECONDS field as an integer value,
     * or 0 if not present.
     *
     * This method works just like {@link #getYears()} except
     * that this method works on the SECONDS field.
     *
     * @return seconds in the integer value. The fraction of seconds
     *   will be discarded (for example, if the actual value is 2.5,
     *   this method returns 2)
     */
    public int getSeconds() {
        return getField(DatatypeConstants.SECONDS).intValue();
    }

    /**
     * <p>Returns the length of the duration in milli-seconds.</p>
     *
     * <p>If the seconds field carries more digits than milli-second order,
     * those will be simply discarded (or in other words, rounded to zero.)
     * For example, for any Calendar value <code>x</code>,</p>
     * <pre>
     * <code>new Duration("PT10.00099S").getTimeInMills(x) == 10000</code>.
     * <code>new Duration("-PT10.00099S").getTimeInMills(x) == -10000</code>.
     * </pre>
     *
     * <p>
     * Note that this method uses the {@link #addTo(Calendar)} method,
     * which may work incorrectly with <code>Duration</code> objects with
     * very large values in its fields. See the {@link #addTo(Calendar)}
     * method for details.
     *
     * @param startInstant
     *      The length of a month/year varies. The <code>startInstant</code> is
     *      used to disambiguate this variance. Specifically, this method
     *      returns the difference between <code>startInstant</code> and
     *      <code>startInstant+duration</code>
     *
     * @return milliseconds between <code>startInstant</code> and
     *   <code>startInstant</code> plus this <code>Duration</code>
     *
     * @throws NullPointerException if <code>startInstant</code> parameter
     * is null.
     *
     */
    public long getTimeInMillis(final Calendar startInstant) {
        Calendar cal = (Calendar) startInstant.clone();
        addTo(cal);
        return getCalendarTimeInMillis(cal)
                    - getCalendarTimeInMillis(startInstant);
    }

    /**
     * <p>Returns the length of the duration in milli-seconds.</p>
```

```
     *
     * <p>If the seconds field carries more digits than milli-second order,
     * those will be simply discarded (or in other words, rounded to zero.)
     * For example, for any <code>Date</code> value <code>x</code>,</p>
     * <pre>
     * <code>new Duration("PT10.00099S").getTimeInMills(x) == 10000</code>.
     * <code>new Duration("-PT10.00099S").getTimeInMills(x) == -10000</code>.
     * </pre>
     *
     * <p>
     * Note that this method uses the {@link #addTo(Date)} method,
     * which may work incorrectly with <code>Duration</code> objects with
     * very large values in its fields. See the {@link #addTo(Date)}
     * method for details.
     *
     * @param startInstant
     *      The length of a month/year varies. The <code>startInstant</code> is
     *      used to disambiguate this variance. Specifically, this method
     *      returns the difference between <code>startInstant</code> and
     *      <code>startInstant+duration</code>.
     *
     * @throws NullPointerException
     *      If the startInstant parameter is null.
     *
     * @return milliseconds between <code>startInstant</code> and
     *   <code>startInstant</code> plus this <code>Duration</code>
     *
     * @see #getTimeInMillis(Calendar)
     */
    public long getTimeInMillis(final Date startInstant) {
        Calendar cal = new GregorianCalendar();
        cal.setTime(startInstant);
        this.addTo(cal);
        return getCalendarTimeInMillis(cal) - startInstant.getTime();
    }

    /**
     * Gets the value of a field.
     *
     * Fields of a duration object may contain arbitrary large value.
     * Therefore this method is designed to return a {@link Number} object.
     *
     * In case of YEARS, MONTHS, DAYS, HOURS, and MINUTES, the returned
     * number will be a non-negative integer. In case of seconds,
     * the returned number may be a non-negative decimal value.
     *
     * @param field
     *      one of the six Field constants (YEARS,MONTHS,DAYS,HOURS,
     *      MINUTES, or SECONDS.)
     * @return
     *      If the specified field is present, this method returns
     *      a non-null non-negative {@link Number} object that
     *      represents its value. If it is not present, return null.
```

```
     *      For YEARS, MONTHS, DAYS, HOURS, and MINUTES, this method
     *      returns a {@link java.math.BigInteger} object. For SECONDS, this
     *      method returns a {@link java.math.BigDecimal}.
     *
     * @throws NullPointerException If the <code>field</code> is <code>null</code>.
     */
    public abstract Number getField(final DatatypeConstants.Field field);

    /**
     * Checks if a field is set.
     *
     * A field of a duration object may or may not be present.
     * This method can be used to test if a field is present.
     *
     * @param field
     *      one of the six Field constants (YEARS,MONTHS,DAYS,HOURS,
     *      MINUTES, or SECONDS.)
     * @return
     *      true if the field is present. false if not.
     *
     * @throws NullPointerException
     *      If the field parameter is null.
     */
    public abstract boolean isSet(final DatatypeConstants.Field field);

    /**
     * <p>Computes a new duration whose value is <code>this+rhs</code>.</p>
     *
     * <p>For example,</p>
     * <pre>
     * "1 day" + "-3 days" = "-2 days"
     * "1 year" + "1 day" = "1 year and 1 day"
     * "-(1 hour,50 minutes)" + "-20 minutes" = "-(1 hours,70 minutes)"
     * "15 hours" + "-3 days" = "-(2 days,9 hours)"
     * "1 year" + "-1 day" = IllegalStateException
     * </pre>
     *
     * <p>Since there's no way to meaningfully subtract 1 day from 1 month,
     * there are cases where the operation fails in
     * {@link IllegalStateException}.</p>
     *
     * <p>
     * Formally, the computation is defined as follows.</p>
     * <p>
     * Firstly, we can assume that two <code>Duration</code>s to be added
     * are both positive without losing generality (i.e.,
     * <code>(-X)+Y=Y-X</code>, <code>X+(-Y)=X-Y</code>,
     * <code>(-X)+(-Y)=-(X+Y)</code>)
     *
     * <p>
     * Addition of two positive <code>Duration</code>s are simply defined as
     * field by field addition where missing fields are treated as 0.
     * <p>
```

```
     * A field of the resulting <code>Duration</code> will be unset if and
     * only if respective fields of two input <code>Duration</code>s are unset.
     * <p>
     * Note that <code>lhs.add(rhs)</code> will be always successful if
     * <code>lhs.signum()*rhs.signum()!=-1</code> or both of them are
     * normalized.</p>
     *
     * @param rhs <code>Duration</code> to add to this <code>Duration</code>
     *
     * @return
     *      non-null valid Duration object.
     *
     * @throws NullPointerException
     *      If the rhs parameter is null.
     * @throws IllegalStateException
     *      If two durations cannot be meaningfully added. For
     *      example, adding negative one day to one month causes
     *      this exception.
     *
     *
     * @see #subtract(Duration)
     */
    public abstract Duration add(final Duration rhs);

    /**
     * Adds this duration to a {@link Calendar} object.
     *
     * <p>
     * Calls {@link java.util.Calendar#add(int,int)} in the
     * order of YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, and MILLISECONDS
     * if those fields are present. Because the {@link Calendar} class
     * uses int to hold values, there are cases where this method
     * won't work correctly (for example if values of fields
     * exceed the range of int.)
     * </p>
     *
     * <p>
     * Also, since this duration class is a Gregorian duration, this
     * method will not work correctly if the given {@link Calendar}
     * object is based on some other calendar systems.
     * </p>
     *
     * <p>
     * Any fractional parts of this <code>Duration</code> object
     * beyond milliseconds will be simply ignored. For example, if
     * this duration is "P1.23456S", then 1 is added to SECONDS,
     * 234 is added to MILLISECONDS, and the rest will be unused.
     * </p>
     *
     * <p>
     * Note that because {@link Calendar#add(int, int)} is using
     * <code>int</code>, <code>Duration</code> with values beyond the
     * range of <code>int</code> in its fields
```

```
     * will cause overflow/underflow to the given {@link Calendar}.
     * {@link XMLGregorianCalendar#add(Duration)} provides the same
     * basic operation as this method while avoiding
     * the overflow/underflow issues.
     *
     * @param calendar
     *      A calendar object whose value will be modified.
     * @throws NullPointerException
     *      if the calendar parameter is null.
     */
    public abstract void addTo(Calendar calendar);

    /**
     * Adds this duration to a {@link Date} object.
     *
     * <p>
     * The given date is first converted into
     * a {@link java.util.GregorianCalendar}, then the duration
     * is added exactly like the {@link #addTo(Calendar)} method.
     *
     * <p>
     * The updated time instant is then converted back into a
     * {@link Date} object and used to update the given {@link Date} object.
     *
     * <p>
     * This somewhat redundant computation is necessary to unambiguously
     * determine the duration of months and years.
     *
     * @param date
     *      A date object whose value will be modified.
     * @throws NullPointerException
     *      if the date parameter is null.
     */
    public void addTo(Date date) {

        // check data parameter
        if (date == null) {
            throw new NullPointerException(
                "Cannot call "
                + this.getClass().getName()
                + "#addTo(Date date) with date == null."
            );
        }

        Calendar cal = new GregorianCalendar();
        cal.setTime(date);
        this.addTo(cal);
        date.setTime(getCalendarTimeInMillis(cal));
    }

    /**
     * <p>Computes a new duration whose value is <code>this-rhs</code>.</p>
     *
```

```
     * <p>For example:</p>
     * <pre>
     * "1 day" - "-3 days" = "4 days"
     * "1 year" - "1 day" = IllegalStateException
     * "-(1 hour,50 minutes)" - "-20 minutes" = "-(1hours,30 minutes)"
     * "15 hours" - "-3 days" = "3 days and 15 hours"
     * "1 year" - "-1 day" = "1 year and 1 day"
     * </pre>
     *
     * <p>Since there's no way to meaningfully subtract 1 day from 1 month,
     * there are cases where the operation fails in {@link IllegalStateException}.</p>
     *
     * <p>Formally the computation is defined as follows.
     * First, we can assume that two <code>Duration</code>s are both positive
     * without losing generality.  (i.e.,
     * <code>(-X)-Y=-(X+Y)</code>, <code>X-(-Y)=X+Y</code>,
     * <code>(-X)-(-Y)=-(X-Y)</code>)</p>
     *
     * <p>Then two durations are subtracted field by field.
     * If the sign of any non-zero field <code>F</code> is different from
     * the sign of the most significant field,
     * 1 (if <code>F</code> is negative) or -1 (otherwise)
     * will be borrowed from the next bigger unit of <code>F</code>.</p>
     *
     * <p>This process is repeated until all the non-zero fields have
     * the same sign.</p>
     *
     * <p>If a borrow occurs in the days field (in other words, if
     * the computation needs to borrow 1 or -1 month to compensate
     * days), then the computation fails by throwing an
     * {@link IllegalStateException}.</p>
     *
     * @param rhs <code>Duration</code> to subtract from this <code>Duration</code>.
     *
     * @return New <code>Duration</code> created from subtracting <code>rhs</code> from this <code>
         Duration</code>.
     *
     * @throws IllegalStateException
     *      If two durations cannot be meaningfully subtracted. For
     *      example, subtracting one day from one month causes
     *      this exception.
     *
     * @throws NullPointerException
     *      If the rhs parameter is null.
     *
     * @see #add(Duration)
     */
    public Duration subtract(final Duration rhs) {
        return add(rhs.negate());
    }

    /**
     * <p>Computes a new duration whose value is <code>factor</code> times
```

```
     * longer than the value of this duration.</p>
     *
     * <p>This method is provided for the convenience.
     * It is functionally equivalent to the following code:</p>
     * <pre>
     * multiply(new BigDecimal(String.valueOf(factor)))
     * </pre>
     *
     * @param factor Factor times longer of new <code>Duration</code> to create.
     *
     * @return New <code>Duration</code> that is <code>factor</code>times longer than this <code>
         Duration</code>.
     *
     * @see #multiply(BigDecimal)
     */
    public Duration multiply(int factor) {
        return multiply(new BigDecimal(String.valueOf(factor)));
    }

    /**
     * Computes a new duration whose value is <code>factor</code> times
     * longer than the value of this duration.
     *
     * <p>
     * For example,
     * <pre>
     * "P1M" (1 month) * "12" = "P12M" (12 months)
     * "PT1M" (1 min) * "0.3" = "PT18S" (18 seconds)
     * "P1M" (1 month) * "1.5" = IllegalStateException
     * </pre>
     *
     * <p>
     * Since the <code>Duration</code> class is immutable, this method
     * doesn't change the value of this object. It simply computes
     * a new Duration object and returns it.
     *
     * <p>
     * The operation will be performed field by field with the precision
     * of {@link BigDecimal}. Since all the fields except seconds are
     * restricted to hold integers,
     * any fraction produced by the computation will be
     * carried down toward the next lower unit. For example,
     * if you multiply "P1D" (1 day) with "0.5", then it will be 0.5 day,
     * which will be carried down to "PT12H" (12 hours).
     * When fractions of month cannot be meaningfully carried down
     * to days, or year to months, this will cause an
     * {@link IllegalStateException} to be thrown.
     * For example if you multiple one month by 0.5.</p>
     *
     * <p>
     * To avoid {@link IllegalStateException}, use
     * the {@link #normalizeWith(Calendar)} method to remove the years
     * and months fields.
```

```
     *
     * @param factor to multiply by
     *
     * @return
     *      returns a non-null valid <code>Duration</code> object
     *
     * @throws IllegalStateException if operation produces fraction in
     * the months field.
     *
     * @throws NullPointerException if the <code>factor</code> parameter is
     * <code>null</code>.
     *
     */
    public abstract Duration multiply(final BigDecimal factor);

    /**
     * Returns a new <code>Duration</code> object whose
     * value is <code>-this</code>.
     *
     * <p>
     * Since the <code>Duration</code> class is immutable, this method
     * doesn't change the value of this object. It simply computes
     * a new Duration object and returns it.
     *
     * @return
     *      always return a non-null valid <code>Duration</code> object.
     */
    public abstract Duration negate();

    /**
     * <p>Converts the years and months fields into the days field
     * by using a specific time instant as the reference point.</p>
     *
     * <p>For example, duration of one month normalizes to 31 days
     * given the start time instance "July 8th 2003, 17:40:32".</p>
     *
     * <p>Formally, the computation is done as follows:</p>
     * <ol>
     *  <li>the given Calendar object is cloned</li>
     *  <li>the years, months and days fields will be added to the {@link Calendar} object
     *      by using the {@link Calendar#add(int,int)} method</li>
     *  <li>the difference between the two Calendars in computed in milliseconds and converted to days,
     *     if a remainder occurs due to Daylight Savings Time, it is discarded</li>
     *  <li>the computed days, along with the hours, minutes and seconds
     *      fields of this duration object is used to construct a new
     *      Duration object.</li>
     * </ol>
     *
     * <p>Note that since the Calendar class uses <code>int</code> to
     * hold the value of year and month, this method may produce
     * an unexpected result if this duration object holds
     * a very large value in the years or months fields.</p>
```

```
     *
     * @param startTimeInstant <code>Calendar</code> reference point.
     *
     * @return <code>Duration</code> of years and months of this <code>Duration</code> as days.
     *
     * @throws NullPointerException If the startTimeInstant parameter is null.
     */
    public abstract Duration normalizeWith(final Calendar startTimeInstant);

    /**
     * <p>Partial order relation comparison with this <code>Duration</code> instance.</p>
     *
     * <p>Comparison result must be in accordance with
     * <a href="http://www.w3.org/TR/xmlschema-2/#duration-order">W3C XML Schema 1.0 Part 2,
       Section 3.2.7.6.2,
     * <i>Order relation on duration</i></a>.</p>
     *
     * <p>Return:</p>
     * <ul>
     *   <li>{@link DatatypeConstants#LESSER} if this <code>Duration</code> is shorter than <code>
       duration</code> parameter</li>
     *   <li>{@link DatatypeConstants#EQUAL} if this <code>Duration</code> is equal to <code>
       duration</code> parameter</li>
     *   <li>{@link DatatypeConstants#GREATER} if this <code>Duration</code> is longer than <code>
       duration</code> parameter</li>
     *   <li>{@link DatatypeConstants#INDETERMINATE} if a conclusive partial order relation cannot
       be determined</li>
     * </ul>
     *
     * @param duration to compare
     *
     * @return the relationship between <code>this</code> <code>Duration</code>and <code>duration</
       code> parameter as
     *   {@link DatatypeConstants#LESSER}, {@link DatatypeConstants#EQUAL}, {@link
       DatatypeConstants#GREATER}
     *   or {@link DatatypeConstants#INDETERMINATE}.
     *
     * @throws UnsupportedOperationException If the underlying implementation
     *   cannot reasonably process the request, e.g. W3C XML Schema allows for
     *   arbitrarily large/small/precise values, the request may be beyond the
     *   implementations capability.
     * @throws NullPointerException if <code>duration</code> is <code>null</code>.
     *
     * @see #isShorterThan(Duration)
     * @see #isLongerThan(Duration)
     */
    public abstract int compare(final Duration duration);

    /**
     * <p>Checks if this duration object is strictly longer than
     * another <code>Duration</code> object.</p>
     *
     * <p>Duration X is "longer" than Y if and only if X>Y
```

```
     * as defined in the section 3.2.6.2 of the XML Schema 1.0
     * specification.</p>
     *
     * <p>For example, "P1D" (one day) > "PT12H" (12 hours) and
     * "P2Y" (two years) > "P23M" (23 months).</p>
     *
     * @param duration <code>Duration</code> to test this <code>Duration</code> against.
     *
     * @throws UnsupportedOperationException If the underlying implementation
     *   cannot reasonably process the request, e.g. W3C XML Schema allows for
     *   arbitrarily large/small/precise values, the request may be beyond the
     *   implementations capability.
     * @throws NullPointerException If <code>duration</code> is null.
     *
     * @return
     *      true if the duration represented by this object
     *      is longer than the given duration. false otherwise.
     *
     * @see #isShorterThan(Duration)
     * @see #compare(Duration duration)
     */
    public boolean isLongerThan(final Duration duration) {
        return compare(duration) == DatatypeConstants.GREATER;
    }

    /**
     * <p>Checks if this duration object is strictly shorter than
     * another <code>Duration</code> object.</p>
     *
     * @param duration <code>Duration</code> to test this <code>Duration</code> against.
     *
     * @return <code>true</code> if <code>duration</code> parameter is shorter than this <code>
         Duration</code>,
     *   else <code>false</code>.
     *
     * @throws UnsupportedOperationException If the underlying implementation
     *   cannot reasonably process the request, e.g. W3C XML Schema allows for
     *   arbitrarily large/small/precise values, the request may be beyond the
     *   implementations capability.
     * @throws NullPointerException if <code>duration</code> is null.
     *
     * @see #isLongerThan(Duration duration)
     * @see #compare(Duration duration)
     */
    public boolean isShorterThan(final Duration duration) {
        return compare(duration) == DatatypeConstants.LESSER;
    }

    /**
     * <p>Checks if this duration object has the same duration
     * as another <code>Duration</code> object.</p>
     *
     * <p>For example, "P1D" (1 day) is equal to "PT24H" (24 hours).</p>
```

```
     *
     * <p>Duration X is equal to Y if and only if time instant
     * t+X and t+Y are the same for all the test time instants
     * specified in the section 3.2.6.2 of the XML Schema 1.0
     * specification.</p>
     *
     * <p>Note that there are cases where two <code>Duration</code>s are
     * "incomparable" to each other, like one month and 30 days.
     * For example,</p>
     * <pre>
     * !new Duration("P1M").isShorterThan(new Duration("P30D"))
     * !new Duration("P1M").isLongerThan(new Duration("P30D"))
     * !new Duration("P1M").equals(new Duration("P30D"))
     * </pre>
     *
     * @param duration
     *      The object to compare this <code>Duration</code> against.
     *
     * @return
     *      <code>true</code> if this duration is the same length as
     *         <code>duration</code>.
     *      <code>false</code> if <code>duration</code> is <code>null</code>,
     *         is not a
     *         <code>Duration</code> object,
     *         or its length is different from this duration.
     *
     * @throws UnsupportedOperationException If the underlying implementation
     *   cannot reasonably process the request, e.g. W3C XML Schema allows for
     *   arbitrarily large/small/precise values, the request may be beyond the
     *   implementations capability.
     *
     * @see #compare(Duration duration)
     */
    public boolean equals(final Object duration) {

        if (duration == null || !(duration instanceof Duration)) {
            return false;
        }

        return compare((Duration) duration) == DatatypeConstants.EQUAL;
    }

    /**
     * Returns a hash code consistent with the definition of the equals method.
     *
     * @see Object#hashCode()
     */
    public abstract int hashCode();

    /**
     * <p>Returns a <code>String</code> representation of this <code>Duration</code> <code>Object</
         code>.</p>
     *
```

```
     * <p>The result is formatted according to the XML Schema 1.0 spec and can be always parsed back later into the
     * equivalent <code>Duration</code> <code>Object</code> by {@link DatatypeFactory#newDuration(String lexicalRepresentation)}.</p>
     *
     * <p>Formally, the following holds for any <code>Duration</code>
     * <code>Object</code> x:</p>
     * <pre>
     * new Duration(x.toString()).equals(x)
     * </pre>
     *
     * @return A non-<code>null</code> valid <code>String</code> representation of this <code>Duration</code>.
     */
    public String toString() {

        StringBuffer buf = new StringBuffer();

        if (getSign() < 0) {
            buf.append('-');
        }
        buf.append('P');

        BigInteger years = (BigInteger) getField(DatatypeConstants.YEARS);
        if (years != null) {
            buf.append(years + "Y");
        }

        BigInteger months = (BigInteger) getField(DatatypeConstants.MONTHS);
        if (months != null) {
            buf.append(months + "M");
        }

        BigInteger days = (BigInteger) getField(DatatypeConstants.DAYS);
        if (days != null) {
            buf.append(days + "D");
        }

        BigInteger hours = (BigInteger) getField(DatatypeConstants.HOURS);
        BigInteger minutes = (BigInteger) getField(DatatypeConstants.MINUTES);
        BigDecimal seconds = (BigDecimal) getField(DatatypeConstants.SECONDS);
        if (hours != null || minutes != null || seconds != null) {
            buf.append('T');
            if (hours != null) {
                buf.append(hours + "H");
            }
            if (minutes != null) {
                buf.append(minutes + "M");
            }
            if (seconds != null) {
                buf.append(toString(seconds) + "S");
            }
        }
```

```java

        return buf.toString();
    }

    /**
     * <p>Turns {@link BigDecimal} to a string representation.</p>
     *
     * <p>Due to a behavior change in the {@link BigDecimal#toString()}
     * method in JDK1.5, this had to be implemented here.</p>
     *
     * @param bd <code>BigDecimal</code> to format as a <code>String</code>
     *
     * @return  <code>String</code> representation of <code>BigDecimal</code>
     */
    private String toString(BigDecimal bd) {
        String intString = bd.unscaledValue().toString();
        int scale = bd.scale();

        if (scale == 0) {
            return intString;
        }

        /* Insert decimal point */
        StringBuffer buf;
        int insertionPoint = intString.length() - scale;
        if (insertionPoint == 0) { /* Point goes right before intVal */
            return "0." + intString;
        } else if (insertionPoint > 0) { /* Point goes inside intVal */
            buf = new StringBuffer(intString);
            buf.insert(insertionPoint, '.');
        } else { /* We must insert zeros between point and intVal */
            buf = new StringBuffer(3 - insertionPoint + intString.length());
            buf.append("0.");
            for (int i = 0; i < -insertionPoint; i++) {
                buf.append('0');
            }
            buf.append(intString);
        }
        return buf.toString();
    }


    /**
     * <p>Calls the {@link Calendar#getTimeInMillis} method.
     * Prior to JDK1.4, this method was protected and therefore
     * cannot be invoked directly.</p>
     *
     * <p>TODO: In future, this should be replaced by <code>cal.getTimeInMillis()</code>.</p>
     *
     * @param cal <code>Calendar</code> to get time in milliseconds.
     *
     * @return Milliseconds of <code>cal</code>.
     */
```

```
    private static long getCalendarTimeInMillis(final Calendar cal) {
        return cal.getTime().getTime();
    }
}
```

14.5 FactoryFinder.java

Secuencia 149: javax/xml/datatype/FactoryFinder.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import java.io.File;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Iterator;
import java.util.Properties;
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * <p>Implements pluggable Datatypes.</p>
 *
 * <p>This class is duplicated for each JAXP subpackage so keep it in
 * sync.  It is package private for secure class loading.</p>
 *
 * @author Santiago.PericasGeertsen@sun.com
 */
class FactoryFinder {
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xerces.internal";
```

```java

    /**
     * Internal debug flag.
     */
    private static boolean debug = false;

    /**
     * Cache for properties in java.home/lib/jaxp.properties
     */
    private final static Properties cacheProps = new Properties();

    /**
     * Flag indicating if properties from java.home/lib/jaxp.properties
     * have been cached.
     */
    private static volatile boolean firstTime = true;

    /**
     * Security support class use to check access control before
     * getting certain system resources.
     */
    private final static SecuritySupport ss = new SecuritySupport();

    // Define system property "jaxp.debug" to get output
    static {
        // Use try/catch block to support applets, which throws
        // SecurityException out of this code.
        try {
            String val = ss.getSystemProperty("jaxp.debug");
            // Allow simply setting the prop to turn on debug
            debug = val != null && !"false".equals(val);
        }
        catch (SecurityException se) {
            debug = false;
        }
    }

    private static void dPrint(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * Attempt to load a class using the class loader supplied. If that fails
     * and fall back is enabled, the current (i.e. bootstrap) class loader is
     * tried.
     *
     * If the class loader supplied is <code>null</code>, first try using the
     * context class loader followed by the current (i.e. bootstrap) class
     * loader.
     *
     * Use bootstrap classLoader if cl = null and useBSClsLoader is true
```

```
     */
    static private Class<?> getProviderClass(String className, ClassLoader cl,
            boolean doFallback, boolean useBSClsLoader) throws ClassNotFoundException
    {
        try {
            if (cl == null) {
                if (useBSClsLoader) {
                    return Class.forName(className, false, FactoryFinder.class.getClassLoader());
                } else {
                    cl = ss.getContextClassLoader();
                    if (cl == null) {
                        throw new ClassNotFoundException();
                    }
                    else {
                        return Class.forName(className, false, cl);
                    }
                }
            }
            else {
                return Class.forName(className, false, cl);
            }
        }
        catch (ClassNotFoundException e1) {
            if (doFallback) {
                // Use current class loader - should always be bootstrap CL
                return Class.forName(className, false, FactoryFinder.class.getClassLoader());
            }
            else {
                throw e1;
            }
        }
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl, boolean doFallback)
        throws  DatatypeConfigurationException
    {
        return newInstance(type, className, cl, doFallback, false);
```

```
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl ClassLoader to use to load the class, null means to use
     * the bootstrap ClassLoader
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     *
     * @param useBSClsLoader True if cl=null actually meant bootstrap classLoader. This parameter
     * is needed since DocumentBuilderFactory/SAXParserFactory defined null as context classLoader.
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl,
            boolean doFallback, boolean useBSClsLoader)
        throws DatatypeConfigurationException
    {
        assert type != null;

        // make sure we have access to restricted packages
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                cl = null;
                useBSClsLoader = true;
            }
        }

        try {
            Class<?> providerClass = getProviderClass(className, cl, doFallback, useBSClsLoader);
            if (!type.isAssignableFrom(providerClass)) {
                throw new ClassCastException(className + " cannot be cast to " + type.getName());
            }
            Object instance = providerClass.newInstance();
            if (debug) {    // Extra check to avoid computing cl strings
                dPrint("created new instance of " + providerClass +
                       " using ClassLoader: " + cl);
            }
            return type.cast(instance);
        }
        catch (ClassNotFoundException x) {
            throw new DatatypeConfigurationException(
                "Provider " + className + " not found", x);
        }
        catch (Exception x) {
            throw new DatatypeConfigurationException(
```

```
                "Provider " + className + " could not be instantiated: " + x,
                x);
        }
    }

    /**
     * Finds the implementation Class object in the specified order.  Main
     * entry point.
     * @return Class object of factory, never null
     *
     * @param type                  Base class / Service interface  of the
     *                              factory to find.
     * @param fallbackClassName     Implementation class name, if nothing else
     *                              is found.  Use null to mean no fallback.
     *
     * Package private so this code can be shared.
     */
    static <T> T find(Class<T> type, String fallbackClassName)
        throws DatatypeConfigurationException
    {
        final String factoryId = type.getName();
        dPrint("find factoryId =" + factoryId);

        // Use the system property first
        try {
            String systemProp = ss.getSystemProperty(factoryId);
            if (systemProp != null) {
                dPrint("found system property, value=" + systemProp);
                return newInstance(type, systemProp, null, true);
            }
        }
        catch (SecurityException se) {
            if (debug) se.printStackTrace();
        }

        // try to read from $java.home/lib/jaxp.properties
        try {
            if (firstTime) {
                synchronized (cacheProps) {
                    if (firstTime) {
                        String configFile = ss.getSystemProperty("java.home") + File.separator +
                            "lib" + File.separator + "jaxp.properties";
                        File f = new File(configFile);
                        firstTime = false;
                        if (ss.doesFileExist(f)) {
                            dPrint("Read properties file "+f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                    }
                }
            }
            final String factoryClassName = cacheProps.getProperty(factoryId);

```

```java
            if (factoryClassName != null) {
                dPrint("found in $java.home/jaxp.properties, value=" + factoryClassName);
                return newInstance(type, factoryClassName, null, true);
            }
        }
        catch (Exception ex) {
            if (debug) ex.printStackTrace();
        }

        // Try Jar Service Provider Mechanism
        final T provider = findServiceProvider(type);
        if (provider != null) {
            return provider;
        }
        if (fallbackClassName == null) {
            throw new DatatypeConfigurationException(
                "Provider for " + factoryId + " cannot be found");
        }

        dPrint("loaded from fallback value: " + fallbackClassName);
        return newInstance(type, fallbackClassName, null, true);
    }

    /*
     * Try to find provider using the ServiceLoader API
     *
     * @param type Base class / Service interface  of the factory to find.
     *
     * @return instance of provider class if found or null
     */
    private static <T> T findServiceProvider(final Class<T> type)
            throws DatatypeConfigurationException
    {
        try {
            return AccessController.doPrivileged(new PrivilegedAction<T>() {
                public T run() {
                    final ServiceLoader<T> serviceLoader = ServiceLoader.load(type);
                    final Iterator<T> iterator = serviceLoader.iterator();
                    if (iterator.hasNext()) {
                        return iterator.next();
                    } else {
                        return null;
                    }
                }
            });
        } catch(ServiceConfigurationError e) {
            final DatatypeConfigurationException error =
                    new DatatypeConfigurationException(
                        "Provider for " + type + " cannot be found", e);
            throw error;
        }
    }

```

```
}
```

14.6 SecuritySupport.java

Secuencia 150: javax/xml/datatype/SecuritySupport.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import java.security.*;
import java.net.*;
import java.io.*;
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


    ClassLoader getContextClassLoader() {
        return (ClassLoader)
                AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                try {
```

```
                    cl = Thread.currentThread().getContextClassLoader();
                } catch (SecurityException ex) { }
                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                    final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
```

```
                }
            })).booleanValue();
    }

}
```

14.7 XMLGregorianCalendar.java

Secuencia 151: javax/xml/datatype/XMLGregorianCalendar.java

```
/*
 * Copyright (c) 2003, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.datatype;

import javax.xml.namespace.QName;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.TimeZone;
import java.util.GregorianCalendar;

/**
 * <p>Representation for W3C XML Schema 1.0 date/time datatypes.
 * Specifically, these date/time datatypes are
 * {@link DatatypeConstants#DATETIME},
 * {@link DatatypeConstants#TIME},
 * {@link DatatypeConstants#DATE},
 * {@link DatatypeConstants#GYEARMONTH},
 * {@link DatatypeConstants#GMONTHDAY},
 * {@link DatatypeConstants#GYEAR},
 * {@link DatatypeConstants#GMONTH}, and
 * {@link DatatypeConstants#GDAY}
```

```
* defined in the XML Namespace
* <code>"http://www.w3.org/2001/XMLSchema"</code>.
* These datatypes are normatively defined in
* <a href="http://www.w3.org/TR/xmlschema-2/#dateTime">W3C XML Schema 1.0 Part 2, Section 3.2.7-14</a>.</p>
*
* <p>The table below defines the mapping between XML Schema 1.0
* date/time datatype fields and this class' fields. It also summarizes
* the value constraints for the date and time fields defined in
* <a href="http://www.w3.org/TR/xmlschema-2/#isoformats">W3C XML Schema 1.0 Part 2, Appendix D,
* <i>ISO 8601 Date and Time Formats</i></a>.</p>
*
* <a name="datetimefieldmapping"/>
* <table border="2" rules="all" cellpadding="2">
*   <thead>
*     <tr>
*       <th align="center" colspan="3">
*         Date/Time Datatype Field Mapping Between XML Schema 1.0 and Java Representation
*       </th>
*     </tr>
*   </thead>
*   <tbody>
*     <tr>
*       <th>XML Schema 1.0<br/>
*           datatype<br/>
*            field</th>
*       <th>Related<br/>XMLGregorianCalendar<br/>Accessor(s)</th>
*       <th>Value Range</th>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-year"/>year</td>
*       <td> {@link #getYear()} + {@link #getEon()} or<br/>
*            {@link #getEonAndYear}
*       </td>
*       <td> <code>getYear()</code> is a value between -(10^9-1) to (10^9)-1
*            or {@link DatatypeConstants#FIELD_UNDEFINED}.<br/>
*            {@link #getEon()} is high order year value in billion of years.<br/>
*            <code>getEon()</code> has values greater than or equal to (10^9) or less than or equal to -(10^9).
*            A value of null indicates field is undefined.</br>
*            Given that <a href="http://www.w3.org/2001/05/xmlschema-errata#e2-63">XML Schema 1.0 errata</a> states that the year zero
*            will be a valid lexical value in a future version of XML Schema,
*            this class allows the year field to be set to zero. Otherwise,
*            the year field value is handled exactly as described
*            in the errata and [ISO-8601-1988]. Note that W3C XML Schema 1.0
*            validation does not allow for the year field to have a value of zero.
*            </td>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-month"/>month</td>
*       <td> {@link #getMonth()} </td>
*       <td> 1 to 12 or {@link DatatypeConstants#FIELD_UNDEFINED} </td>
```

```
*     </tr>
*     <tr>
*       <td><a name="datetimefield-day"/>day</td>
*       <td> {@link #getDay()} </td>
*       <td> Independent of month, max range is 1 to 31 or {@link DatatypeConstants#FIELD_UNDEFINED}.<br/>
*           The normative value constraint stated relative to month
*           field's value is in <a href="http://www.w3.org/TR/xmlschema-2/#isoformats">W3C XML Schema 1.0 Part 2, Appendix D</a>.
*       </td>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-hour"/>hour</td>
*       <td>{@link #getHour()}</td>
*       <td>
*         0 to 23 or {@link DatatypeConstants#FIELD_UNDEFINED}.
*         An hour value of 24 is allowed to be set in the lexical space provided the minute and second
*         field values are zero. However, an hour value of 24 is not allowed in value space and will be
*         transformed to represent the value of the first instance of the following day as per
*         <a href="http://www.w3.org/TR/xmlschema-2/#built-in-primitive-datatypes">
*         XML Schema Part 2: Datatypes Second Edition, 3.2 Primitive datatypes</a>.
*       </td>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-minute"/>minute</td>
*       <td> {@link #getMinute()} </td>
*       <td> 0 to 59 or {@link DatatypeConstants#FIELD_UNDEFINED} </td>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-second"/>second</td>
*       <td>
*         {@link #getSecond()} + {@link #getMillisecond()}/1000 or<br/>
*         {@link #getSecond()} + {@link #getFractionalSecond()}
*       </td>
*       <td>
*         {@link #getSecond()} from 0 to 60 or {@link DatatypeConstants#FIELD_UNDEFINED}.<br/>
*         <i>(Note: 60 only allowable for leap second.)</i><br/>
*         {@link #getFractionalSecond()} allows for infinite precision over the range from 0.0 to 1.0 when
*         the {@link #getSecond()} is defined.<br/>
*         <code>FractionalSecond</code> is optional and has a value of <code>null</code> when it is undefined.<br />
*           {@link #getMillisecond()} is the convenience
*           millisecond precision of value of {@link #getFractionalSecond()}.
*       </td>
*     </tr>
*     <tr>
*       <td><a name="datetimefield-timezone"/>timezone</td>
*       <td> {@link #getTimezone()} </td>
*       <td> Number of minutes or {@link DatatypeConstants#FIELD_UNDEFINED}.
*         Value range from -14 hours (-14 * 60 minutes) to 14 hours (14 * 60 minutes).
```

```
 *       </td>
 *     </tr>
 *   </tbody>
 *  </table>
 *
 * <p>All maximum value space constraints listed for the fields in the table
 * above are checked by factory methods, @{link DatatypeFactory},
 * setter methods and parse methods of
 * this class. <code>IllegalArgumentException</code> is thrown when a
 * parameter's value is outside the value constraint for the field or
 * if the composite
 * values constitute an invalid XMLGregorianCalendar instance (for example, if
 * the 31st of June is specified).
 * </p>
 *
 * <p>The following operations are defined for this class:
 * <ul>
 *   <li>accessors/mutators for independent date/time fields</li>
 *   <li>conversion between this class and W3C XML Schema 1.0 lexical representation,
 *     {@link #toString()}, {@link DatatypeFactory#newXMLGregorianCalendar(String lexicalRepresentation)}</li>
 *   <li>conversion between this class and {@link GregorianCalendar},
 *     {@link #toGregorianCalendar(java.util.TimeZone timezone, java.util.Locale aLocale, XMLGregorianCalendar defaults)},
 *     {@link DatatypeFactory}</li>
 *   <li>partial order relation comparator method, {@link #compare(XMLGregorianCalendar xmlGregorianCalendar)}</li>
 *   <li>{@link #equals(Object)} defined relative to {@link #compare(XMLGregorianCalendar xmlGregorianCalendar)}.</li>
 *   <li>addition operation with {@link Duration}
 *      instance as defined in <a href="http://www.w3.org/TR/xmlschema-2/#adding-durations-to-dateTimes">
 *      W3C XML Schema 1.0 Part 2, Appendix E, <i>Adding durations to dateTimes</i></a>.
 *   </li>
 * </ul>
 * </p>
 *
 * @author <a href="mailto:Joseph.Fialli@Sun.com">Joseph Fialli</a>
 * @author <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Sunitha.Reddy@Sun.com">Sunitha Reddy</a>
 * @see Duration
 * @see DatatypeFactory
 * @since 1.5
 */

public abstract class XMLGregorianCalendar
        implements Cloneable {

        /**
         * Default no-arg constructor.
         *
         * <p>Note: Always use the {@link DatatypeFactory} to
```

```
     * construct an instance of <code>XMLGregorianCalendar</code>.
     * The constructor on this class cannot be guaranteed to
     * produce an object with a consistent state and may be
     * removed in the future.</p>
     */
     public XMLGregorianCalendar() {
     }

    /**
     * <p>Unset all fields to undefined.</p>
     *
     * <p>Set all int fields to {@link DatatypeConstants#FIELD_UNDEFINED} and reference fields
     * to null.</p>
     */
    public abstract void clear();

    /**
     * <p>Reset this <code>XMLGregorianCalendar</code> to its original values.</p>
     *
     * <p><code>XMLGregorianCalendar</code> is reset to the same values as when it was created
         with
     * {@link DatatypeFactory#newXMLGregorianCalendar()},
     * {@link DatatypeFactory#newXMLGregorianCalendar(String lexicalRepresentation)},
     * {@link DatatypeFactory#newXMLGregorianCalendar(
     *   BigInteger year,
     *   int month,
     *   int day,
     *   int hour,
     *   int minute,
     *   int second,
     *   BigDecimal fractionalSecond,
     *   int timezone)},
     * {@link DatatypeFactory#newXMLGregorianCalendar(
     *   int year,
     *   int month,
     *   int day,
     *   int hour,
     *   int minute,
     *   int second,
     *   int millisecond,
     *   int timezone)},
     * {@link DatatypeFactory#newXMLGregorianCalendar(GregorianCalendar cal)},
     * {@link DatatypeFactory#newXMLGregorianCalendarDate(
     *   int year,
     *   int month,
     *   int day,
     *   int timezone)},
     * {@link DatatypeFactory#newXMLGregorianCalendarTime(
     *   int hours,
     *   int minutes,
     *   int seconds,
     *   int timezone)},
     * {@link DatatypeFactory#newXMLGregorianCalendarTime(
```

```
        *   int hours,
        *   int minutes,
        *   int seconds,
        *   BigDecimal fractionalSecond,
        *   int timezone)} or
        * {@link DatatypeFactory#newXMLGregorianCalendarTime(
        *   int hours,
        *   int minutes,
        *   int seconds,
        *   int milliseconds,
        *   int timezone)}.
        * </p>
        *
        * <p><code>reset()</code> is designed to allow the reuse of existing <code>XMLGregorianCalendar</code>s
        * thus saving resources associated with the creation of new <code>XMLGregorianCalendar</code>s.</p>
        */
       public abstract void reset();

    /**
     * <p>Set low and high order component of XSD <code>dateTime</code> year field.</p>
     *
     * <p>Unset this field by invoking the setter with a parameter value of <code>null</code>.</p>
     *
     * @param year value constraints summarized in <a href="#datetimefield-year">year field of date/time field mapping table</a>.
     *
     * @throws IllegalArgumentException if <code>year</code> parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public abstract void setYear(BigInteger year);

    /**
     * <p>Set year of XSD <code>dateTime</code> year field.</p>
     *
     * <p>Unset this field by invoking the setter with a parameter value of
     * {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
     *
     * <p>Note: if the absolute value of the <code>year</code> parameter
     * is less than 10^9, the eon component of the XSD year field is set to
     * <code>null</code> by this method.</p>
     *
     * @param year value constraints are summarized in <a href="#datetimefield-year">year field of date/time field mapping table</a>.
     *   If year is {@link DatatypeConstants#FIELD_UNDEFINED}, then eon is set to <code>null</code>.
     */
    public abstract void setYear(int year);

    /**
     * <p>Set month.</p>
```

```
     *
     * <p>Unset this field by invoking the setter with a parameter value of {@link
         DatatypeConstants#FIELD_UNDEFINED}.</p>
     *
     * @param month value constraints summarized in <a href="#datetimefield-month">month field of
         date/time field mapping table</a>.
     *
     * @throws IllegalArgumentException if <code>month</code> parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public abstract void setMonth(int month);

    /**
     * <p>Set days in month.</p>
     *
     * <p>Unset this field by invoking the setter with a parameter value of {@link
         DatatypeConstants#FIELD_UNDEFINED}.</p>
     *
     * @param day value constraints summarized in <a href="#datetimefield-day">day field of date/
         time field mapping table</a>.
     *
     * @throws IllegalArgumentException if <code>day</code> parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public abstract void setDay(int day);

    /**
     * <p>Set the number of minutes in the timezone offset.</p>
     *
     * <p>Unset this field by invoking the setter with a parameter value of {@link
         DatatypeConstants#FIELD_UNDEFINED}.</p>
     *
     * @param offset value constraints summarized in <a href="#datetimefield-timezone">
     *   timezone field of date/time field mapping table</a>.
     *
     * @throws IllegalArgumentException if <code>offset</code> parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public abstract void setTimezone(int offset);

    /**
     * <p>Set time as one unit.</p>
     *
     * @param hour value constraints are summarized in
     * <a href="#datetimefield-hour">hour field of date/time field mapping table</a>.
     * @param minute value constraints are summarized in
     * <a href="#datetimefield-minute">minute field of date/time field mapping table</a>.
     * @param second value constraints are summarized in
     * <a href="#datetimefield-second">second field of date/time field mapping table</a>.
     *
```

```
     * @see #setTime(int, int, int, BigDecimal)
     *
     * @throws IllegalArgumentException if any parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public void setTime(int hour, int minute, int second) {

                setTime(
                        hour,
                        minute,
                        second,
                        null // fractional
                );
    }

        /**
         * <p>Set hours.</p>
         *
         * <p>Unset this field by invoking the setter with a parameter value of {@link
             DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * @param hour value constraints summarized in <a href="#datetimefield-hour">hour field of
             date/time field mapping table</a>.
         *
         * @throws IllegalArgumentException if <code>hour</code> parameter is outside value
             constraints for the field as specified in
         *   <a href="#datetimefieldmapping">date/time field mapping table</a>.
        */
        public abstract void setHour(int hour);

        /**
         * <p>Set minutes.</p>
         *
         * <p>Unset this field by invoking the setter with a parameter value of {@link
             DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * @param minute value constraints summarized in <a href="#datetimefield-minute">minute
             field of date/time field mapping table</a>.
         *
         * @throws IllegalArgumentException if <code>minute</code> parameter is outside value
             constraints for the field as specified in
         *   <a href="#datetimefieldmapping">date/time field mapping table</a>.
        */
        public abstract void setMinute(int minute);

        /**
         * <p>Set seconds.</p>
         *
         * <p>Unset this field by invoking the setter with a parameter value of {@link
             DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * @param second value constraints summarized in <a href="#datetimefield-second">second
```

```
            field of date/time field mapping table</a>.
         *
         * @throws IllegalArgumentException if <code>second</code> parameter is outside value constraints for the field as specified in
         *   <a href="#datetimefieldmapping">date/time field mapping table</a>.
        */
        public abstract void setSecond(int second);

        /**
         * <p>Set milliseconds.</p>
         *
         * <p>Unset this field by invoking the setter with a parameter value of {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * @param millisecond value constraints summarized in
         *   <a href="#datetimefield-second">second field of date/time field mapping table</a>.
         *
         * @throws IllegalArgumentException if <code>millisecond</code> parameter is outside value constraints for the field as specified
         *   in <a href="#datetimefieldmapping">date/time field mapping table</a>.
        */
        public abstract void setMillisecond(int millisecond);

        /**
         * <p>Set fractional seconds.</p>
         *
         * <p>Unset this field by invoking the setter with a parameter value of <code>null</code>.</p>
         *
         * @param fractional value constraints summarized in
         *   <a href="#datetimefield-second">second field of date/time field mapping table</a>.
         *
         * @throws IllegalArgumentException if <code>fractional</code> parameter is outside value constraints for the field as specified
         *   in <a href="#datetimefieldmapping">date/time field mapping table</a>.
        */
        public abstract void setFractionalSecond(BigDecimal fractional);


    /**
     * <p>Set time as one unit, including the optional infinite precision
     * fractional seconds.</p>
     *
     * @param hour value constraints are summarized in
     * <a href="#datetimefield-hour">hour field of date/time field mapping table</a>.
     * @param minute value constraints are summarized in
     * <a href="#datetimefield-minute">minute field of date/time field mapping table</a>.
     * @param second value constraints are summarized in
     * <a href="#datetimefield-second">second field of date/time field mapping table</a>.
     * @param fractional value of <code>null</code> indicates this optional
     *   field is not set.
     *
     * @throws IllegalArgumentException if any parameter is
```

```
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public void setTime(
        int hour,
        int minute,
        int second,
        BigDecimal fractional) {

                setHour(hour);
        setMinute(minute);
        setSecond(second);
        setFractionalSecond(fractional);
    }


    /**
     * <p>Set time as one unit, including optional milliseconds.</p>
     *
     * @param hour value constraints are summarized in
     * <a href="#datetimefield-hour">hour field of date/time field mapping table</a>.
     * @param minute value constraints are summarized in
     * <a href="#datetimefield-minute">minute field of date/time field mapping table</a>.
     * @param second value constraints are summarized in
     * <a href="#datetimefield-second">second field of date/time field mapping table</a>.
     * @param millisecond value of {@link DatatypeConstants#FIELD_UNDEFINED} indicates this
     *                    optional field is not set.
     *
     * @throws IllegalArgumentException if any parameter is
     * outside value constraints for the field as specified in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.
     */
    public void setTime(int hour, int minute, int second, int millisecond) {

        setHour(hour);
        setMinute(minute);
        setSecond(second);
        setMillisecond(millisecond);
    }

        /**
         * <p>Return high order component for XML Schema 1.0 dateTime datatype field for
         * <code>year</code>.
         * <code>null</code> if this optional part of the year field is not defined.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-year">year field of date/time field mapping table</a>.</p>
         * @return eon of this <code>XMLGregorianCalendar</code>. The value
         * returned is an integer multiple of 10^9.
         *
         * @see #getYear()
         * @see #getEonAndYear()
         */
```

```java
        public abstract BigInteger getEon();

        /**
         * <p>Return low order component for XML Schema 1.0 dateTime datatype field for
         * <code>year</code> or {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-year">year field of date/time field mapping table</a>.</p>
         *
         * @return year  of this <code>XMLGregorianCalendar</code>.
         *
         * @see #getEon()
         * @see #getEonAndYear()
         */
        public abstract int getYear();

        /**
         * <p>Return XML Schema 1.0 dateTime datatype field for
         * <code>year</code>.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-year">year field of date/time field mapping table</a>.</p>
         *
         * @return sum of <code>eon</code> and <code>BigInteger.valueOf(year)</code>
         * when both fields are defined. When only <code>year</code> is defined,
         * return it. When both <code>eon</code> and <code>year</code> are not
         * defined, return <code>null</code>.
         *
         * @see #getEon()
         * @see #getYear()
         */
        public abstract BigInteger getEonAndYear();

        /**
         * <p>Return number of month or {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-month">month field of date/time field mapping table</a>.</p>
         *
         * @return year  of this <code>XMLGregorianCalendar</code>.
         *
         */
        public abstract int getMonth();

        /**
         * Return day in month or {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-day">day field of date/time field mapping table</a>.</p>
         *
         * @see #setDay(int)
         */
        public abstract int getDay();
```

```
537
538         /**
539          * Return timezone offset in minutes or
540          * {@link DatatypeConstants#FIELD_UNDEFINED} if this optional field is not defined.
541          *
542          * <p>Value constraints for this value are summarized in
543          * <a href="#datetimefield-timezone">timezone field of date/time field mapping table</a>.</
               p>
544          *
545          * @see #setTimezone(int)
546          */
547         public abstract int getTimezone();
548
549         /**
550          * Return hours or {@link DatatypeConstants#FIELD_UNDEFINED}.
551          * Returns {@link DatatypeConstants#FIELD_UNDEFINED} if this field is not defined.
552          *
553          * <p>Value constraints for this value are summarized in
554          * <a href="#datetimefield-hour">hour field of date/time field mapping table</a>.</p>
555          * @see #setTime(int, int, int)
556          */
557         public abstract int getHour();
558
559         /**
560          * Return minutes or {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
561          * Returns {@link DatatypeConstants#FIELD_UNDEFINED} if this field is not defined.
562          *
563          * <p>Value constraints for this value are summarized in
564          * <a href="#datetimefield-minute">minute field of date/time field mapping table</a>.</p>
565          * @see #setTime(int, int, int)
566          */
567         public abstract int getMinute();
568
569         /**
570          * <p>Return seconds or {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
571          *
572          * <p>Returns {@link DatatypeConstants#FIELD_UNDEFINED} if this field is not defined.
573          * When this field is not defined, the optional xs:dateTime
574          * fractional seconds field, represented by
575          * {@link #getFractionalSecond()} and {@link #getMillisecond()},
576          * must not be defined.</p>
577          *
578          * <p>Value constraints for this value are summarized in
579          * <a href="#datetimefield-second">second field of date/time field mapping table</a>.</p>
580          *
581          * @return Second  of this <code>XMLGregorianCalendar</code>.
582          *
583          * @see #getFractionalSecond()
584          * @see #getMillisecond()
585          * @see #setTime(int, int, int)
586          */
587         public abstract int getSecond();
588
```

```java
        /**
         * <p>Return millisecond precision of {@link #getFractionalSecond()}.</p>
         *
         * <p>This method represents a convenience accessor to infinite
         * precision fractional second value returned by
         * {@link #getFractionalSecond()}. The returned value is the rounded
         * down to milliseconds value of
         * {@link #getFractionalSecond()}. When {@link #getFractionalSecond()}
         * returns <code>null</code>, this method must return
         * {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * <p>Value constraints for this value are summarized in
         * <a href="#datetimefield-second">second field of date/time field mapping table</a>.</p>
         *
         * @return Millisecond  of this <code>XMLGregorianCalendar</code>.
         *
         * @see #getFractionalSecond()
         * @see #setTime(int, int, int)
         */
        public int getMillisecond() {

                BigDecimal fractionalSeconds = getFractionalSecond();

                // is field undefined?
                if (fractionalSeconds == null) {
                        return DatatypeConstants.FIELD_UNDEFINED;
                }

                return getFractionalSecond().movePointRight(3).intValue();
        }

        /**
         * <p>Return fractional seconds.</p>
         *
         * <p><code>null</code> is returned when this optional field is not defined.</p>
         *
         * <p>Value constraints are detailed in
         * <a href="#datetimefield-second">second field of date/time field mapping table</a>.</p>
         *
         * <p>This optional field can only have a defined value when the
         * xs:dateTime second field, represented by {@link #getSecond()},
         * does not return {@link DatatypeConstants#FIELD_UNDEFINED}.</p>
         *
         * @return fractional seconds  of this <code>XMLGregorianCalendar</code>.
         *
         * @see #getSecond()
         * @see #setTime(int, int, int, BigDecimal)
         */
        public abstract BigDecimal getFractionalSecond();

    // comparisons
    /**
     * <p>Compare two instances of W3C XML Schema 1.0 date/time datatypes
```

```
     * according to partial order relation defined in
     * <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">W3C XML Schema 1.0 Part 2,
       Section 3.2.7.3,
     * <i>Order relation on dateTime</i></a>.</p>
     *
     * <p><code>xsd:dateTime</code> datatype field mapping to accessors of
     * this class are defined in
     * <a href="#datetimefieldmapping">date/time field mapping table</a>.</p>
     *
     * @param xmlGregorianCalendar Instance of <code>XMLGregorianCalendar</code> to compare
     *
     * @return The relationship between <code>this</code> <code>XMLGregorianCalendar</code> and
     *   the specified <code>xmlGregorianCalendar</code> as
     *   {@link DatatypeConstants#LESSER},
     *   {@link DatatypeConstants#EQUAL},
     *   {@link DatatypeConstants#GREATER} or
     *   {@link DatatypeConstants#INDETERMINATE}.
     *
     * @throws NullPointerException if <code>xmlGregorianCalendar</code> is null.
     */
    public abstract int compare(XMLGregorianCalendar xmlGregorianCalendar);

    /**
     * <p>Normalize this instance to UTC.</p>
     *
     * <p>2000-03-04T23:00:00+03:00 normalizes to 2000-03-04T20:00:00Z</p>
     * <p>Implements W3C XML Schema Part 2, Section 3.2.7.3 (A).</p>
     *
     * @return <code>this</code> <code>XMLGregorianCalendar</code> normalized to UTC.
     */
    public abstract XMLGregorianCalendar normalize();

    /**
     * <p>Compares this calendar to the specified object. The result is
     * <code>true</code> if and only if the argument is not null and is an
     * <code>XMLGregorianCalendar</code> object that represents the same
     * instant in time as this object.</p>
     *
     * @param obj to compare.
     *
     * @return <code>true</code> when <code>obj</code> is an instance of
     * <code>XMLGregorianCalendar</code> and
     * {@link #compare(XMLGregorianCalendar obj)}
     * returns {@link DatatypeConstants#EQUAL},
     * otherwise <code>false</code>.
     */
    public boolean equals(Object obj) {

        if (obj == null || !(obj instanceof XMLGregorianCalendar)) {
               return false;
        }
        return compare((XMLGregorianCalendar) obj) == DatatypeConstants.EQUAL;
    }
```

```java

    /**
     * <p>Returns a hash code consistent with the definition of the equals method.</p>
     *
     * @return hash code of this object.
     */
    public int hashCode() {

        // Following two dates compare to EQUALS since in different timezones.
        // 2000-01-15T12:00:00-05:00 == 2000-01-15T13:00:00-04:00
        //
        // Must ensure both instances generate same hashcode by normalizing
        // this to UTC timezone.
        int timezone = getTimezone();
        if (timezone == DatatypeConstants.FIELD_UNDEFINED) {
            timezone = 0;
        }
        XMLGregorianCalendar gc = this;
        if (timezone != 0) {
            gc = this.normalize();
        }
        return gc.getYear()
                + gc.getMonth()
                + gc.getDay()
                + gc.getHour()
                + gc.getMinute()
                + gc.getSecond();
    }

    /**
     * <p>Return the lexical representation of <code>this</code> instance.
     * The format is specified in
     * <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">XML Schema 1.0 Part 2, Section 3.2.[7-14].1,
     * <i>Lexical Representation</i>".</a></p>
     *
     * <p>Specific target lexical representation format is determined by
     * {@link #getXMLSchemaType()}.</p>
     *
     * @return XML, as <code>String</code>, representation of this <code>XMLGregorianCalendar</code>
     *
     * @throws IllegalStateException if the combination of set fields
     *    does not match one of the eight defined XML Schema builtin date/time datatypes.
     */
    public abstract String toXMLFormat();

    /**
     * <p>Return the name of the XML Schema date/time type that this instance
     * maps to. Type is computed based on fields that are set.</p>
     *
     * <table border="2" rules="all" cellpadding="2">
     *   <thead>
```

```
     *     <tr>
     *       <th align="center" colspan="7">
     *         Required fields for XML Schema 1.0 Date/Time Datatypes.<br/>
     *         <i>(timezone is optional for all date/time datatypes)</i>
     *       </th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td>Datatype</td>
     *       <td>year</td>
     *       <td>month</td>
     *       <td>day</td>
     *       <td>hour</td>
     *       <td>minute</td>
     *       <td>second</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#DATETIME}</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#DATE}</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#TIME}</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td>X</td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#GYEARMONTH}</td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
```

```
     *     <tr>
     *       <td>{@link DatatypeConstants#GMONTHDAY}</td>
     *       <td></td>
     *       <td>X</td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#GYEAR}</td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#GMONTH}</td>
     *       <td></td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *     <tr>
     *       <td>{@link DatatypeConstants#GDAY}</td>
     *       <td></td>
     *       <td></td>
     *       <td>X</td>
     *       <td></td>
     *       <td></td>
     *       <td></td>
     *     </tr>
     *   </tbody>
     * </table>
     *
     * @throws java.lang.IllegalStateException if the combination of set fields
     *    does not match one of the eight defined XML Schema builtin
     *    date/time datatypes.
     * @return One of the following class constants:
     *   {@link DatatypeConstants#DATETIME},
     *   {@link DatatypeConstants#TIME},
     *   {@link DatatypeConstants#DATE},
     *   {@link DatatypeConstants#GYEARMONTH},
     *   {@link DatatypeConstants#GMONTHDAY},
     *   {@link DatatypeConstants#GYEAR},
     *   {@link DatatypeConstants#GMONTH} or
     *   {@link DatatypeConstants#GDAY}.
     */
    public abstract QName getXMLSchemaType();
```

```
        /**
         * <p>Returns a <code>String</code> representation of this <code>XMLGregorianCalendar</code>
             > <code>Object</code>.</p>
         *
         * <p>The result is a lexical representation generated by {@link #toXMLFormat()}.</p>
         *
         * @return A non-<code>null</code> valid <code>String</code> representation of this <code>
             XMLGregorianCalendar</code>.
         *
     * @throws IllegalStateException if the combination of set fields
     *    does not match one of the eight defined XML Schema builtin date/time datatypes.
     *
     * @see #toXMLFormat()
         */
    public String toString() {

        return toXMLFormat();
    }

    /**
     * Validate instance by <code>getXMLSchemaType()</code> constraints.
     * @return true if data values are valid.
     */
    public abstract boolean isValid();

    /**
     * <p>Add <code>duration</code> to this instance.</p>
     *
     * <p>The computation is specified in
     * <a href="http://www.w3.org/TR/xmlschema-2/#adding-durations-to-dateTimes">XML Schema 1.0
         Part 2, Appendix E,
     * <i>Adding durations to dateTimes</i>></a>.
     * <a href="#datetimefieldmapping">date/time field mapping table</a>
     * defines the mapping from XML Schema 1.0 <code>dateTime</code> fields
     * to this class' representation of those fields.</p>
     *
     * @param duration Duration to add to this <code>XMLGregorianCalendar</code>.
     *
     * @throws NullPointerException  when <code>duration</code> parameter is <code>null</code>.
     */
    public abstract void add(Duration duration);

    /**
     * <p>Convert this <code>XMLGregorianCalendar</code> to a {@link GregorianCalendar}.</p>
     *
     * <p>When <code>this</code> instance has an undefined field, this
     * conversion relies on the <code>java.util.GregorianCalendar</code> default
     * for its corresponding field. A notable difference between
     * XML Schema 1.0 date/time datatypes and <code>java.util.GregorianCalendar</code>
     * is that Timezone value is optional for date/time datatypes and it is
     * a required field for <code>java.util.GregorianCalendar</code>. See javadoc
     * for <code>java.util.TimeZone.getDefault()</code> on how the default
```

```
     * is determined. To explicitly specify the <code>TimeZone</code>
     * instance, see
     * {@link #toGregorianCalendar(TimeZone, Locale, XMLGregorianCalendar)}.</p>
     *
     * <table border="2" rules="all" cellpadding="2">
     *   <thead>
     *     <tr>
     *       <th align="center" colspan="2">
     *          Field by Field Conversion from this class to
     *          <code>java.util.GregorianCalendar</code>
     *       </th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *        <td><code>java.util.GregorianCalendar</code> field</td>
     *        <td><code>javax.xml.datatype.XMLGregorianCalendar</code> field</td>
     *     </tr>
     *     <tr>
     *       <td><code>ERA</code></td>
     *       <td>{@link #getEonAndYear()}<code>.signum() < 0 ? GregorianCalendar.BC :
         GregorianCalendar.AD</code></td>
     *     </tr>
     *     <tr>
     *       <td><code>YEAR</code></td>
     *       <td>{@link #getEonAndYear()}<code>.abs().intValue()</code><i>*</i></td>
     *     </tr>
     *     <tr>
     *       <td><code>MONTH</code></td>
     *       <td>{@link #getMonth()} - {@link DatatypeConstants#JANUARY} + {@link GregorianCalendar
         #JANUARY}</td>
     *     </tr>
     *     <tr>
     *       <td><code>DAY_OF_MONTH</code></td>
     *       <td>{@link #getDay()}</td>
     *     </tr>
     *     <tr>
     *       <td><code>HOUR_OF_DAY</code></td>
     *       <td>{@link #getHour()}</td>
     *     </tr>
     *     <tr>
     *       <td><code>MINUTE</code></td>
     *       <td>{@link #getMinute()}</td>
     *     </tr>
     *     <tr>
     *       <td><code>SECOND</code></td>
     *       <td>{@link #getSecond()}</td>
     *     </tr>
     *     <tr>
     *       <td><code>MILLISECOND</code></td>
     *       <td>get millisecond order from {@link #getFractionalSecond()}<i>*</i> </td>
     *     </tr>
     *     <tr>
```

```
     *       <td><code>GregorianCalendar.setTimeZone(TimeZone)</code></td>
     *       <td>{@link #getTimezone()} formatted into Custom timezone id</td>
     *     </tr>
     *   </tbody>
     * </table>
     * <i>*</i> designates possible loss of precision during the conversion due
     * to source datatype having higher precision than target datatype.
     *
     * <p>To ensure consistency in conversion implementations, the new
     * <code>GregorianCalendar</code> should be instantiated in following
     * manner.
     * <ul>
     *   <li>Using <code>timeZone</code> value as defined above, create a new
     * <code>java.util.GregorianCalendar(timeZone,Locale.getDefault())</code>.
     *   </li>
     *   <li>Initialize all GregorianCalendar fields by calling {@link java.util.GregorianCalendar#
         clear()}.</li>
     *   <li>Obtain a pure Gregorian Calendar by invoking
     *   <code>GregorianCalendar.setGregorianChange(
     *   new Date(Long.MIN_VALUE))</code>.</li>
     *   <li>Its fields ERA, YEAR, MONTH, DAY_OF_MONTH, HOUR_OF_DAY,
     *       MINUTE, SECOND and MILLISECOND are set using the method
     *       <code>Calendar.set(int,int)</code></li>
     * </ul>
     * </p>
     *
     * @see #toGregorianCalendar(java.util.TimeZone, java.util.Locale, XMLGregorianCalendar)
     */
    public abstract GregorianCalendar toGregorianCalendar();

    /**
     * <p>Convert this <code>XMLGregorianCalendar</code> along with provided parameters
     * to a {@link GregorianCalendar} instance.</p>
     *
     * <p> Since XML Schema 1.0 date/time datetypes has no concept of
     * timezone ids or daylight savings timezone ids, this conversion operation
     * allows the user to explicitly specify one with
     * <code>timezone</code> parameter.</p>
     *
     * <p>To compute the return value's <code>TimeZone</code> field,
     * <ul>
     * <li>when parameter <code>timeZone</code> is non-null,
     * it is the timezone field.</li>
     * <li>else when <code>this.getTimezone() != FIELD_UNDEFINED</code>,
     * create a <code>java.util.TimeZone</code> with a custom timezone id
     * using the <code>this.getTimezone()</code>.</li>
     * <li>else when <code>defaults.getTimezone() != FIELD_UNDEFINED</code>,
     * create a <code>java.util.TimeZone</code> with a custom timezone id
     * using <code>defaults.getTimezone()</code>.</li>
     * <li>else use the <code>GregorianCalendar</code> default timezone value
     * for the host is defined as specified by
     * <code>java.util.TimeZone.getDefault()</code>.</li></p>
     *
```

```
     * <p>To ensure consistency in conversion implementations, the new
     * <code>GregorianCalendar</code> should be instantiated in following
     * manner.
     * <ul>
     *   <li>Create a new <code>java.util.GregorianCalendar(TimeZone,
     *       Locale)</code> with TimeZone set as specified above and the
     *       <code>Locale</code> parameter.
     *   </li>
     *   <li>Initialize all GregorianCalendar fields by calling {@link GregorianCalendar#clear()}</
         li>
     *   <li>Obtain a pure Gregorian Calendar by invoking
     *   <code>GregorianCalendar.setGregorianChange(
     *   new Date(Long.MIN_VALUE))</code>.</li>
     *   <li>Its fields ERA, YEAR, MONTH, DAY_OF_MONTH, HOUR_OF_DAY,
     *       MINUTE, SECOND and MILLISECOND are set using the method
     *       <code>Calendar.set(int,int)</code></li>
     * </ul>
     *
     * @param timezone provide Timezone. <code>null</code> is a legal value.
     * @param aLocale  provide explicit Locale. Use default GregorianCalendar locale if
     *                 value is <code>null</code>.
     * @param defaults provide default field values to use when corresponding
     *                 field for this instance is FIELD_UNDEFINED or null.
     *                 If <code>defaults</code>is <code>null</code> or a field
     *                 within the specified <code>defaults</code> is undefined,
     *                 just use <code>java.util.GregorianCalendar</code> defaults.
     * @return a java.util.GregorianCalendar conversion of this instance.
     */
    public abstract GregorianCalendar toGregorianCalendar(
        java.util.TimeZone timezone,
                java.util.Locale aLocale,
                XMLGregorianCalendar defaults);

    /**
     * <p>Returns a <code>java.util.TimeZone</code> for this class.</p>
     *
     * <p>If timezone field is defined for this instance,
     * returns TimeZone initialized with custom timezone id
     * of zoneoffset. If timezone field is undefined,
     * try the defaultZoneoffset that was passed in.
     * If defaultZoneoffset is FIELD_UNDEFINED, return
     * default timezone for this host.
     * (Same default as java.util.GregorianCalendar).</p>
     *
     * @param defaultZoneoffset default zoneoffset if this zoneoffset is
     * {@link DatatypeConstants#FIELD_UNDEFINED}.
     *
     * @return TimeZone for this.
     */
    public abstract TimeZone getTimeZone(int defaultZoneoffset);
```

```
    /**
     * <p>Creates and returns a copy of this object.</p>
     *
     * @return copy of this <code>Object</code>
     */
   public abstract Object clone();
}
```

15 Xml Namespace (javax.xml.namespace package)

15.1 NamespaceContext.java

Secuencia 152: javax/xml/namespace/NamespaceContext.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.namespace;

import java.util.Iterator;

/**
 * <p>Interface for read only XML Namespace context processing.</p>
 *
 * <p>An XML Namespace has the properties:</p>
 * <ul>
 *   <li>Namespace URI:
 *       Namespace name expressed as a URI to which the prefix is bound</li>
 *   <li>prefix: syntactically, this is the part of the attribute name
 *       following the <code>XMLConstants.XMLNS_ATTRIBUTE</code>
 *       ("xmlns") in the Namespace declaration</li>
 * </ul>
```

```
 * <p>example:
 * <code>&lt;element xmlns:prefix="http://Namespace-name-URI"&gt;</code></p>
 *
 * <p>All <code>get*(*)</code> methods operate in the current scope
 * for Namespace URI and prefix resolution.</p>
 *
 * <p>Note that a Namespace URI can be bound to
 * <strong>multiple</strong> prefixes in the current scope.  This can
 * occur when multiple <code>XMLConstants.XMLNS_ATTRIBUTE</code>
 * ("xmlns") Namespace declarations occur in the same Start-Tag and
 * refer to the same Namespace URI. e.g.<br />
 * <pre>
 * &lt;element xmlns:prefix1="http://Namespace-name-URI"
 *          xmlns:prefix2="http://Namespace-name-URI"&gt;
 * </pre>
 * This can also occur when the same Namespace URI is used in multiple
 * <code>XMLConstants.XMLNS_ATTRIBUTE</code> ("xmlns") Namespace
 * declarations in the logical parent element hierarchy.  e.g.<br />
 * <pre>
 * &lt;parent xmlns:prefix1="http://Namespace-name-URI">
 *   &lt;child xmlns:prefix2="http://Namespace-name-URI"&gt;
 *     ...
 *   &lt;/child&gt;
 * &lt;/parent&gt;
 * </pre></p>
 *
 * <p>A prefix can only be bound to a <strong>single</strong>
 * Namespace URI in the current scope.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @see javax.xml.XMLConstants
 *   javax.xml.XMLConstants for declarations of common XML values
 * @see <a href="http://www.w3.org/TR/xmlschema-2/#QName">
 *   XML Schema Part2: Datatypes</a>
 * @see <a href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
 *   Namespaces in XML</a>
 * @see <a href="http://www.w3.org/XML/xml-names-19990114-errata">
 *   Namespaces in XML Errata</a>
 * @since 1.5
 */

public interface NamespaceContext {

    /**
     * <p>Get Namespace URI bound to a prefix in the current scope.</p>
     *
     * <p>When requesting a Namespace URI by prefix, the following
     * table describes the returned Namespace URI value for all
     * possible prefix values:</p>
     *
     * <table border="2" rules="all" cellpadding="4">
     *   <thead>
     *     <tr>
```

```
     *       <td align="center" colspan="2">
     *         <code>getNamespaceURI(prefix)</code>
     *         return value for specified prefixes
     *       </td>
     *     </tr>
     *     <tr>
     *       <td>prefix parameter</td>
     *       <td>Namespace URI return value</td>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td><code>DEFAULT_NS_PREFIX</code> ("")</td>
     *       <td>default Namespace URI in the current scope or
     *         <code>{@link
     *         javax.xml.XMLConstants#NULL_NS_URI XMLConstants.NULL_NS_URI("")}
     *         </code>
     *         when there is no default Namespace URI in the current scope</td>
     *     </tr>
     *     <tr>
     *       <td>bound prefix</td>
     *       <td>Namespace URI bound to prefix in current scope</td>
     *     </tr>
     *     <tr>
     *       <td>unbound prefix</td>
     *       <td>
     *         <code>{@link
     *         javax.xml.XMLConstants#NULL_NS_URI XMLConstants.NULL_NS_URI("")}
     *         </code>
     *       </td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XML_NS_PREFIX</code> ("xml")</td>
     *       <td><code>XMLConstants.XML_NS_URI</code>
     *           ("http://www.w3.org/XML/1998/namespace")</td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XMLNS_ATTRIBUTE</code> ("xmlns")</td>
     *       <td><code>XMLConstants.XMLNS_ATTRIBUTE_NS_URI</code>
     *         ("http://www.w3.org/2000/xmlns/")</td>
     *     </tr>
     *     <tr>
     *       <td><code>null</code></td>
     *       <td><code>IllegalArgumentException</code> is thrown</td>
     *     </tr>
     *    </tbody>
     * </table>
     *
     * @param prefix prefix to look up
     *
     * @return Namespace URI bound to prefix in the current scope
     *
     * @throws IllegalArgumentException When <code>prefix</code> is
```

```
     *   <code>null</code>
     */
    String getNamespaceURI(String prefix);

    /**
     * <p>Get prefix bound to Namespace URI in the current scope.</p>
     *
     * <p>To get all prefixes bound to a Namespace URI in the current
     * scope, use {@link #getPrefixes(String namespaceURI)}.</p>
     *
     * <p>When requesting a prefix by Namespace URI, the following
     * table describes the returned prefix value for all Namespace URI
     * values:</p>
     *
     * <table border="2" rules="all" cellpadding="4">
     *   <thead>
     *     <tr>
     *       <th align="center" colspan="2">
     *         <code>getPrefix(namespaceURI)</code> return value for
     *         specified Namespace URIs
     *       </th>
     *     </tr>
     *     <tr>
     *       <th>Namespace URI parameter</th>
     *       <th>prefix value returned</th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td>&lt;default Namespace URI&gt;</td>
     *       <td><code>XMLConstants.DEFAULT_NS_PREFIX</code> ("")
     *       </td>
     *     </tr>
     *     <tr>
     *       <td>bound Namespace URI</td>
     *       <td>prefix bound to Namespace URI in the current scope,
     *           if multiple prefixes are bound to the Namespace URI in
     *           the current scope, a single arbitrary prefix, whose
     *           choice is implementation dependent, is returned</td>
     *     </tr>
     *     <tr>
     *       <td>unbound Namespace URI</td>
     *       <td><code>null</code></td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XML_NS_URI</code>
     *           ("http://www.w3.org/XML/1998/namespace")</td>
     *       <td><code>XMLConstants.XML_NS_PREFIX</code> ("xml")</td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XMLNS_ATTRIBUTE_NS_URI</code>
     *           ("http://www.w3.org/2000/xmlns/")</td>
     *       <td><code>XMLConstants.XMLNS_ATTRIBUTE</code> ("xmlns")</td>
```

```
     *     </tr>
     *     <tr>
     *       <td><code>null</code></td>
     *       <td><code>IllegalArgumentException</code> is thrown</td>
     *     </tr>
     *   </tbody>
     * </table>
     *
     * @param namespaceURI URI of Namespace to lookup
     *
     * @return prefix bound to Namespace URI in current context
     *
     * @throws IllegalArgumentException When <code>namespaceURI</code> is
     *   <code>null</code>
     */
    String getPrefix(String namespaceURI);

    /**
     * <p>Get all prefixes bound to a Namespace URI in the current
     * scope.</p>
     *
     * <p>An Iterator over String elements is returned in an arbitrary,
     * <strong>implementation dependent</strong>, order.</p>
     *
     * <p><strong>The <code>Iterator</code> is
     * <em>not</em> modifiable.  e.g. the
     * <code>remove()</code> method will throw
     * <code>UnsupportedOperationException</code>.</strong></p>
     *
     * <p>When requesting prefixes by Namespace URI, the following
     * table describes the returned prefixes value for all Namespace
     * URI values:</p>
     *
     * <table border="2" rules="all" cellpadding="4">
     *   <thead>
     *     <tr>
     *       <th align="center" colspan="2"><code>
     *         getPrefixes(namespaceURI)</code> return value for
     *         specified Namespace URIs</th>
     *     </tr>
     *     <tr>
     *       <th>Namespace URI parameter</th>
     *       <th>prefixes value returned</th>
     *     </tr>
     *   </thead>
     *   <tbody>
     *     <tr>
     *       <td>bound Namespace URI,
     *         including the &lt;default Namespace URI&gt;</td>
     *       <td>
     *         <code>Iterator</code> over prefixes bound to Namespace URI in
     *         the current scope in an arbitrary,
     *         <strong>implementation dependent</strong>,
```

```
     *         order
     *       </td>
     *     </tr>
     *     <tr>
     *       <td>unbound Namespace URI</td>
     *       <td>empty <code>Iterator</code></td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XML_NS_URI</code>
     *           ("http://www.w3.org/XML/1998/namespace")</td>
     *       <td><code>Iterator</code> with one element set to
     *         <code>XMLConstants.XML_NS_PREFIX</code> ("xml")</td>
     *     </tr>
     *     <tr>
     *       <td><code>XMLConstants.XMLNS_ATTRIBUTE_NS_URI</code>
     *           ("http://www.w3.org/2000/xmlns/")</td>
     *       <td><code>Iterator</code> with one element set to
     *         <code>XMLConstants.XMLNS_ATTRIBUTE</code> ("xmlns")</td>
     *     </tr>
     *     <tr>
     *       <td><code>null</code></td>
     *       <td><code>IllegalArgumentException</code> is thrown</td>
     *     </tr>
     *   </tbody>
     * </table>
     *
     * @param namespaceURI URI of Namespace to lookup
     *
     * @return <code>Iterator</code> for all prefixes bound to the
     *   Namespace URI in the current scope
     *
     * @throws IllegalArgumentException When <code>namespaceURI</code> is
     *   <code>null</code>
     */
    Iterator getPrefixes(String namespaceURI);
}
```

15.2 QName.java

Secuencia 153: javax/xml/namespace/QName.java

```
/*
 * Copyright (c) 2003, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.namespace;

import java.io.Serializable;
import java.security.AccessController;
import java.security.PrivilegedAction;

import javax.xml.XMLConstants;

/**
 * <p><code>QName</code> represents a <strong>qualified name</strong>
 * as defined in the XML specifications: <a
 * href="http://www.w3.org/TR/xmlschema-2/#QName">XML Schema Part2:
 * Datatypes specification</a>, <a
 * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">Namespaces
 * in XML</a>, <a
 * href="http://www.w3.org/XML/xml-names-19990114-errata">Namespaces
 * in XML Errata</a>.</p>
 *
 * <p>The value of a <code>QName</code> contains a <strong>Namespace
 * URI</strong>, <strong>local part</strong> and
 * <strong>prefix</strong>.</p>
 *
 * <p>The prefix is included in <code>QName</code> to retain lexical
 * information <strong><em>when present</em></strong> in an {@link
 * javax.xml.transform.Source XML input source}. The prefix is
 * <strong><em>NOT</em></strong> used in {@link #equals(Object)
 * QName.equals(Object)} or to compute the {@link #hashCode()
 * QName.hashCode()}.  Equality and the hash code are defined using
 * <strong><em>only</em></strong> the Namespace URI and local part.</p>
 *
 * <p>If not specified, the Namespace URI is set to {@link
 * javax.xml.XMLConstants#NULL_NS_URI XMLConstants.NULL_NS_URI}.
 * If not specified, the prefix is set to {@link
 * javax.xml.XMLConstants#DEFAULT_NS_PREFIX
 * XMLConstants.DEFAULT_NS_PREFIX}.</p>
 *
 * <p><code>QName</code> is immutable.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @version $Revision: 1.8 $, $Date: 2010/03/18 03:06:17 $
 * @see <a href="http://www.w3.org/TR/xmlschema-2/#QName">
```

```
 *   XML Schema Part2: Datatypes specification</a>
 * @see <a href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">
 *   Namespaces in XML</a>
 * @see <a href="http://www.w3.org/XML/xml-names-19990114-errata">
 *   Namespaces in XML Errata</a>
 * @since 1.5
 */

public class QName implements Serializable {

    /**
     * <p>Stream Unique Identifier.</p>
     *
     * <p>Due to a historical defect, QName was released with multiple
     * serialVersionUID values even though its serialization was the
     * same.</p>
     *
     * <p>To workaround this issue, serialVersionUID is set with either
     * a default value or a compatibility value.  To use the
     * compatiblity value, set the system property:</p>
     *
     * <code>com.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0</code>
     *
     * <p>This workaround was inspired by classes in the javax.management
     * package, e.g. ObjectName, etc.
     * See CR6267224 for original defect report.</p>
     */
    private static final long serialVersionUID;
    /**
     * <p>Default <code>serialVersionUID</code> value.</p>
     */
    private static final long defaultSerialVersionUID = -9120448754896609940L;
    /**
     * <p>Compatibility <code>serialVersionUID</code> value.</p>
     */
    private static final long compatibleSerialVersionUID = 4418622981026545151L;
    /**
     * <p>Flag to use default or campatible serialVersionUID.</p>
     */
    private static boolean useDefaultSerialVersionUID = true;
    static {
        try {
            // use a privileged block as reading a system property
            String valueUseCompatibleSerialVersionUID = (String) AccessController.doPrivileged(
                    new PrivilegedAction() {
                        public Object run() {
                            return System.getProperty("com.sun.xml.namespace.QName.
                                useCompatibleSerialVersionUID");
                        }
                    }
            );
            useDefaultSerialVersionUID = (valueUseCompatibleSerialVersionUID != null &&
                valueUseCompatibleSerialVersionUID.equals("1.0")) ? false : true;
```

```
        } catch (Exception exception) {
            // use default if any Exceptions
            useDefaultSerialVersionUID = true;
        }

        // set serialVersionUID to desired value
        if (useDefaultSerialVersionUID) {
            serialVersionUID = defaultSerialVersionUID;
        } else {
            serialVersionUID = compatibleSerialVersionUID;
        }
    }

    /**
     * <p>Namespace URI of this <code>QName</code>.</p>
     */
    private final String namespaceURI;

    /**
     * <p>local part of this <code>QName</code>.</p>
     */
    private final String localPart;

    /**
     * <p>prefix of this <code>QName</code>.</p>
     */
    private final String prefix;

    /**
     * <p><code>QName</code> constructor specifying the Namespace URI
     * and local part.</p>
     *
     * <p>If the Namespace URI is <code>null</code>, it is set to
     * {@link javax.xml.XMLConstants#NULL_NS_URI
     * XMLConstants.NULL_NS_URI}.  This value represents no
     * explicitly defined Namespace as defined by the <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">Namespaces
     * in XML</a> specification.  This action preserves compatible
     * behavior with QName 1.0.  Explicitly providing the {@link
     * javax.xml.XMLConstants#NULL_NS_URI
     * XMLConstants.NULL_NS_URI} value is the preferred coding
     * style.</p>
     *
     * <p>If the local part is <code>null</code> an
     * <code>IllegalArgumentException</code> is thrown.
     * A local part of "" is allowed to preserve
     * compatible behavior with QName 1.0. </p>
     *
     * <p>When using this constructor, the prefix is set to {@link
     * javax.xml.XMLConstants#DEFAULT_NS_PREFIX
     * XMLConstants.DEFAULT_NS_PREFIX}.</p>
     *
     * <p>The Namespace URI is not validated as a
```

```
     * <a href="http://www.ietf.org/rfc/rfc2396.txt">URI reference</a>.
     * The local part is not validated as a
     * <a href="http://www.w3.org/TR/REC-xml-names/#NT-NCName">NCName</a>
     * as specified in <a href="http://www.w3.org/TR/REC-xml-names/">Namespaces
     * in XML</a>.</p>
     *
     * @param namespaceURI Namespace URI of the <code>QName</code>
     * @param localPart    local part of the <code>QName</code>
     *
     * @throws IllegalArgumentException When <code>localPart</code> is
     *   <code>null</code>
     *
     * @see #QName(String namespaceURI, String localPart, String
     * prefix) QName(String namespaceURI, String localPart, String
     * prefix)
     */
    public QName(final String namespaceURI, final String localPart) {
        this(namespaceURI, localPart, XMLConstants.DEFAULT_NS_PREFIX);
    }

    /**
     * <p><code>QName</code> constructor specifying the Namespace URI,
     * local part and prefix.</p>
     *
     * <p>If the Namespace URI is <code>null</code>, it is set to
     * {@link javax.xml.XMLConstants#NULL_NS_URI
     * XMLConstants.NULL_NS_URI}.  This value represents no
     * explicitly defined Namespace as defined by the <a
     * href="http://www.w3.org/TR/REC-xml-names/#ns-qualnames">Namespaces
     * in XML</a> specification.  This action preserves compatible
     * behavior with QName 1.0.  Explicitly providing the {@link
     * javax.xml.XMLConstants#NULL_NS_URI
     * XMLConstants.NULL_NS_URI} value is the preferred coding
     * style.</p>
     *
     * <p>If the local part is <code>null</code> an
     * <code>IllegalArgumentException</code> is thrown.
     * A local part of "" is allowed to preserve
     * compatible behavior with QName 1.0. </p>
     *
     * <p>If the prefix is <code>null</code>, an
     * <code>IllegalArgumentException</code> is thrown.  Use {@link
     * javax.xml.XMLConstants#DEFAULT_NS_PREFIX
     * XMLConstants.DEFAULT_NS_PREFIX} to explicitly indicate that no
     * prefix is present or the prefix is not relevant.</p>
     *
     * <p>The Namespace URI is not validated as a
     * <a href="http://www.ietf.org/rfc/rfc2396.txt">URI reference</a>.
     * The local part and prefix are not validated as a
     * <a href="http://www.w3.org/TR/REC-xml-names/#NT-NCName">NCName</a>
     * as specified in <a href="http://www.w3.org/TR/REC-xml-names/">Namespaces
     * in XML</a>.</p>
     *
```

```
     * @param namespaceURI Namespace URI of the <code>QName</code>
     * @param localPart    local part of the <code>QName</code>
     * @param prefix       prefix of the <code>QName</code>
     *
     * @throws IllegalArgumentException When <code>localPart</code>
     *   or <code>prefix</code> is <code>null</code>
     */
    public QName(String namespaceURI, String localPart, String prefix) {

        // map null Namespace URI to default
        // to preserve compatibility with QName 1.0
        if (namespaceURI == null) {
            this.namespaceURI = XMLConstants.NULL_NS_URI;
        } else {
            this.namespaceURI = namespaceURI;
        }

        // local part is required.
        // "" is allowed to preserve compatibility with QName 1.0
        if (localPart == null) {
            throw new IllegalArgumentException(
                    "local part cannot be \"null\" when creating a QName");
        }
        this.localPart = localPart;

        // prefix is required
        if (prefix == null) {
            throw new IllegalArgumentException(
                    "prefix cannot be \"null\" when creating a QName");
        }
        this.prefix = prefix;
    }

    /**
     * <p><code>QName</code> constructor specifying the local part.</p>
     *
     * <p>If the local part is <code>null</code> an
     * <code>IllegalArgumentException</code> is thrown.
     * A local part of "" is allowed to preserve
     * compatible behavior with QName 1.0. </p>
     *
     * <p>When using this constructor, the Namespace URI is set to
     * {@link javax.xml.XMLConstants#NULL_NS_URI
     * XMLConstants.NULL_NS_URI} and the prefix is set to {@link
     * javax.xml.XMLConstants#DEFAULT_NS_PREFIX
     * XMLConstants.DEFAULT_NS_PREFIX}.</p>
     *
     * <p><em>In an XML context, all Element and Attribute names exist
     * in the context of a Namespace.  Making this explicit during the
     * construction of a <code>QName</code> helps prevent hard to
     * diagnosis XML validity errors.  The constructors {@link
     * #QName(String namespaceURI, String localPart) QName(String
     * namespaceURI, String localPart)} and
```

```
     * {@link #QName(String namespaceURI, String localPart, String prefix)}
     * are preferred.</em></p>
     *
     * <p>The local part is not validated as a
     * <a href="http://www.w3.org/TR/REC-xml-names/#NT-NCName">NCName</a>
     * as specified in <a href="http://www.w3.org/TR/REC-xml-names/">Namespaces
     * in XML</a>.</p>
     *
     * @param localPart local part of the <code>QName</code>
     *
     * @throws IllegalArgumentException When <code>localPart</code> is
     *   <code>null</code>
     *
     * @see #QName(String namespaceURI, String localPart) QName(String
     * namespaceURI, String localPart)
     * @see #QName(String namespaceURI, String localPart, String
     * prefix) QName(String namespaceURI, String localPart, String
     * prefix)
     */
    public QName(String localPart) {
        this(
            XMLConstants.NULL_NS_URI,
            localPart,
            XMLConstants.DEFAULT_NS_PREFIX);
    }

    /**
     * <p>Get the Namespace URI of this <code>QName</code>.</p>
     *
     * @return Namespace URI of this <code>QName</code>
     */
    public String getNamespaceURI() {
        return namespaceURI;
    }

    /**
     * <p>Get the local part of this <code>QName</code>.</p>
     *
     *  @return local part of this <code>QName</code>
     */
    public String getLocalPart() {
        return localPart;
    }

    /**
     * <p>Get the prefix of this <code>QName</code>.</p>
     *
     * <p>The prefix assigned to a <code>QName</code> might
     * <strong><em>NOT</em></strong> be valid in a different
     * context. For example, a <code>QName</code> may be assigned a
     * prefix in the context of parsing a document but that prefix may
     * be invalid in the context of a different document.</p>
     *
```

```
     *  @return prefix of this <code>QName</code>
     */
    public String getPrefix() {
        return prefix;
    }

    /**
     * <p>Test this <code>QName</code> for equality with another
     * <code>Object</code>.</p>
     *
     * <p>If the <code>Object</code> to be tested is not a
     * <code>QName</code> or is <code>null</code>, then this method
     * returns <code>false</code>.</p>
     *
     * <p>Two <code>QName</code>s are considered equal if and only if
     * both the Namespace URI and local part are equal. This method
     * uses <code>String.equals()</code> to check equality of the
     * Namespace URI and local part. The prefix is
     * <strong><em>NOT</em></strong> used to determine equality.</p>
     *
     * <p>This method satisfies the general contract of {@link
     * java.lang.Object#equals(Object) Object.equals(Object)}</p>
     *
     * @param objectToTest the <code>Object</code> to test for
     * equality with this <code>QName</code>
     * @return <code>true</code> if the given <code>Object</code> is
     * equal to this <code>QName</code> else <code>false</code>
     */
    public final boolean equals(Object objectToTest) {
        if (objectToTest == this) {
            return true;
        }

        if (objectToTest == null || !(objectToTest instanceof QName)) {
            return false;
        }

        QName qName = (QName) objectToTest;

        return localPart.equals(qName.localPart)
            && namespaceURI.equals(qName.namespaceURI);
    }

    /**
     * <p>Generate the hash code for this <code>QName</code>.</p>
     *
     * <p>The hash code is calculated using both the Namespace URI and
     * the local part of the <code>QName</code>.  The prefix is
     * <strong><em>NOT</em></strong> used to calculate the hash
     * code.</p>
     *
     * <p>This method satisfies the general contract of {@link
     * java.lang.Object#hashCode() Object.hashCode()}.</p>
```

```
     *
     * @return hash code for this <code>QName</code> <code>Object</code>
     */
    public final int hashCode() {
        return namespaceURI.hashCode() ^ localPart.hashCode();
    }

    /**
     * <p><code>String</code> representation of this
     * <code>QName</code>.</p>
     *
     * <p>The commonly accepted way of representing a <code>QName</code>
     * as a <code>String</code> was
     * <a href="http://jclark.com/xml/xmlns.htm">defined</a>
     * by James Clark.  Although this is not a <em>standard</em>
     * specification, it is in common use, e.g. {@link
     * javax.xml.transform.Transformer#setParameter(String name, Object value)}.
     * This implementation represents a <code>QName</code> as:
     * "{" + Namespace URI + "}" + local part.  If the Namespace URI
     * <code>.equals(XMLConstants.NULL_NS_URI)</code>, only the
     * local part is returned.  An appropriate use of this method is
     * for debugging or logging for human consumption.</p>
     *
     * <p>Note the prefix value is <strong><em>NOT</em></strong>
     * returned as part of the <code>String</code> representation.</p>
     *
     * <p>This method satisfies the general contract of {@link
     * java.lang.Object#toString() Object.toString()}.</p>
     *
     *  @return <code>String</code> representation of this <code>QName</code>
     */
    public String toString() {
        if (namespaceURI.equals(XMLConstants.NULL_NS_URI)) {
            return localPart;
        } else {
            return "{" + namespaceURI + "}" + localPart;
        }
    }

    /**
     * <p><code>QName</code> derived from parsing the formatted
     * <code>String</code>.</p>
     *
     * <p>If the <code>String</code> is <code>null</code> or does not conform to
     * {@link #toString() QName.toString()} formatting, an
     * <code>IllegalArgumentException</code> is thrown.</p>
     *
     * <p><em>The <code>String</code> <strong>MUST</strong> be in the
     * form returned by {@link #toString() QName.toString()}.</em></p>
     *
     * <p>The commonly accepted way of representing a <code>QName</code>
     * as a <code>String</code> was
     * <a href="http://jclark.com/xml/xmlns.htm">defined</a>
```

```
     * by James Clark.  Although this is not a <em>standard</em>
     * specification, it is in common use, e.g. {@link
     * javax.xml.transform.Transformer#setParameter(String name, Object value)}.
     * This implementation parses a <code>String</code> formatted
     * as: "{" + Namespace URI + "}" + local part.  If the Namespace
     * URI <code>.equals(XMLConstants.NULL_NS_URI)</code>, only the
     * local part should be provided.</p>
     *
     * <p>The prefix value <strong><em>CANNOT</em></strong> be
     * represented in the <code>String</code> and will be set to
     * {@link javax.xml.XMLConstants#DEFAULT_NS_PREFIX
     * XMLConstants.DEFAULT_NS_PREFIX}.</p>
     *
     * <p>This method does not do full validation of the resulting
     * <code>QName</code>.
     * <p>The Namespace URI is not validated as a
     * <a href="http://www.ietf.org/rfc/rfc2396.txt">URI reference</a>.
     * The local part is not validated as a
     * <a href="http://www.w3.org/TR/REC-xml-names/#NT-NCName">NCName</a>
     * as specified in
     * <a href="http://www.w3.org/TR/REC-xml-names/">Namespaces in XML</a>.</p>
     *
     * @param qNameAsString <code>String</code> representation
     * of the <code>QName</code>
     *
     * @throws IllegalArgumentException When <code>qNameAsString</code> is
     *   <code>null</code> or malformed
     *
     * @return <code>QName</code> corresponding to the given <code>String</code>
     * @see #toString() QName.toString()
     */
    public static QName valueOf(String qNameAsString) {

        // null is not valid
        if (qNameAsString == null) {
            throw new IllegalArgumentException(
                    "cannot create QName from \"null\" or \"\" String");
        }

        // "" local part is valid to preserve compatible behavior with QName 1.0
        if (qNameAsString.length() == 0) {
            return new QName(
                XMLConstants.NULL_NS_URI,
                qNameAsString,
                XMLConstants.DEFAULT_NS_PREFIX);
        }

        // local part only?
        if (qNameAsString.charAt(0) != '{') {
            return new QName(
                XMLConstants.NULL_NS_URI,
                qNameAsString,
                XMLConstants.DEFAULT_NS_PREFIX);
```

```
        }

        // Namespace URI improperly specified?
        if (qNameAsString.startsWith("{" + XMLConstants.NULL_NS_URI + "}")) {
            throw new IllegalArgumentException(
                "Namespace URI .equals(XMLConstants.NULL_NS_URI), "
                + ".equals(\"" + XMLConstants.NULL_NS_URI + "\"), "
                + "only the local part, "
                + "\""
                + qNameAsString.substring(2 + XMLConstants.NULL_NS_URI.length())
                + "\", "
                + "should be provided.");
        }

        // Namespace URI and local part specified
        int endOfNamespaceURI = qNameAsString.indexOf('}');
        if (endOfNamespaceURI == -1) {
            throw new IllegalArgumentException(
                "cannot create QName from \""
                    + qNameAsString
                    + "\", missing closing \"}\"");
        }
        return new QName(
            qNameAsString.substring(1, endOfNamespaceURI),
            qNameAsString.substring(endOfNamespaceURI + 1),
            XMLConstants.DEFAULT_NS_PREFIX);
    }
}
```

16 Xml Parsers (javax.xml.parsers package)

16.1 DocumentBuilder.java

Secuencia 154: javax/xml/parsers/DocumentBuilder.java

```
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

package javax.xml.parsers;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;

import javax.xml.validation.Schema;

import org.w3c.dom.Document;
import org.w3c.dom.DOMImplementation;

import org.xml.sax.EntityResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

/**
 * Defines the API to obtain DOM Document instances from an XML
 * document. Using this class, an application programmer can obtain a
 * {@link Document} from XML.<p>
 *
 * An instance of this class can be obtained from the
 * {@link DocumentBuilderFactory#newDocumentBuilder()} method. Once
 * an instance of this class is obtained, XML can be parsed from a
 * variety of input sources. These input sources are InputStreams,
 * Files, URLs, and SAX InputSources.<p>
 *
 * Note that this class reuses several classes from the SAX API. This
 * does not require that the implementor of the underlying DOM
 * implementation use a SAX parser to parse XML document into a
 * <code>Document</code>. It merely requires that the implementation
 * communicate with the application using these existing APIs.
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */

public abstract class DocumentBuilder {


    /** Protected constructor */
    protected DocumentBuilder () {
    }

    /**
     * <p>Reset this <code>DocumentBuilder</code> to its original configuration.</p>
     *
     * <p><code>DocumentBuilder</code> is reset to the same state as when it was created with
```

```
     * {@link DocumentBuilderFactory#newDocumentBuilder()}.
     * <code>reset()</code> is designed to allow the reuse of existing <code>DocumentBuilder</code>s
     * thus saving resources associated with the creation of new <code>DocumentBuilder</code>s.</p>
     *
     * <p>The reset <code>DocumentBuilder</code> is not guaranteed to have the same {@link EntityResolver} or {@link ErrorHandler}
     * <code>Object</code>s, e.g. {@link Object#equals(Object obj)}.  It is guaranteed to have a functionally equal
     * <code>EntityResolver</code> and <code>ErrorHandler</code>.</p>
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
     * @since 1.5
     */
    public void reset() {

        // implementors should override this method
        throw new UnsupportedOperationException(
                "This DocumentBuilder, \"" + this.getClass().getName() + "\", does not support the reset functionality."
                + "  Specification \"" + this.getClass().getPackage().getSpecificationTitle() + "\"
                + " version \"" + this.getClass().getPackage().getSpecificationVersion() + "\""
                );
    }

    /**
     * Parse the content of the given <code>InputStream</code> as an XML
     * document and return a new DOM {@link Document} object.
     * An <code>IllegalArgumentException</code> is thrown if the
     * <code>InputStream</code> is null.
     *
     * @param is InputStream containing the content to be parsed.
     *
     * @return <code>Document</code> result of parsing the
     *  <code>InputStream</code>
     *
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any parse errors occur.
     * @throws IllegalArgumentException When <code>is</code> is <code>null</code>
     *
     * @see org.xml.sax.DocumentHandler
     */

    public Document parse(InputStream is)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource in = new InputSource(is);
```

```
        return parse(in);
    }

    /**
     * Parse the content of the given <code>InputStream</code> as an
     * XML document and return a new DOM {@link Document} object.
     * An <code>IllegalArgumentException</code> is thrown if the
     * <code>InputStream</code> is null.
     *
     * @param is InputStream containing the content to be parsed.
     * @param systemId Provide a base for resolving relative URIs.
     *
     * @return A new DOM Document object.
     *
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any parse errors occur.
     * @throws IllegalArgumentException When <code>is</code> is <code>null</code>
     *
     * @see org.xml.sax.DocumentHandler
     */

    public Document parse(InputStream is, String systemId)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource in = new InputSource(is);
        in.setSystemId(systemId);
        return parse(in);
    }

    /**
     * Parse the content of the given URI as an XML document
     * and return a new DOM {@link Document} object.
     * An <code>IllegalArgumentException</code> is thrown if the
     * URI is <code>null</code> null.
     *
     * @param uri The location of the content to be parsed.
     *
     * @return A new DOM Document object.
     *
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any parse errors occur.
     * @throws IllegalArgumentException When <code>uri</code> is <code>null</code>
     *
     * @see org.xml.sax.DocumentHandler
     */

    public Document parse(String uri)
        throws SAXException, IOException {
        if (uri == null) {
            throw new IllegalArgumentException("URI cannot be null");
```

```
        }

        InputSource in = new InputSource(uri);
        return parse(in);
    }

    /**
     * Parse the content of the given file as an XML document
     * and return a new DOM {@link Document} object.
     * An <code>IllegalArgumentException</code> is thrown if the
     * <code>File</code> is <code>null</code> null.
     *
     * @param f The file containing the XML to parse.
     *
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any parse errors occur.
     * @throws IllegalArgumentException When <code>f</code> is <code>null</code>
     *
     * @see org.xml.sax.DocumentHandler
     * @return A new DOM Document object.
     */

    public Document parse(File f) throws SAXException, IOException {
        if (f == null) {
            throw new IllegalArgumentException("File cannot be null");
        }

        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        InputSource in = new InputSource(f.toURI().toASCIIString());
        return parse(in);
    }

    /**
     * Parse the content of the given input source as an XML document
     * and return a new DOM {@link Document} object.
     * An <code>IllegalArgumentException</code> is thrown if the
     * <code>InputSource</code> is <code>null</code> null.
     *
     * @param is InputSource containing the content to be parsed.
     *
     * @return A new DOM Document object.
     *
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any parse errors occur.
     * @throws IllegalArgumentException When <code>is</code> is <code>null</code>
     *
     * @see org.xml.sax.DocumentHandler
     */

    public abstract Document parse(InputSource is)
        throws SAXException, IOException;
```

```


    /**
     * Indicates whether or not this parser is configured to
     * understand namespaces.
     *
     * @return true if this parser is configured to understand
     *         namespaces; false otherwise.
     */

    public abstract boolean isNamespaceAware();

    /**
     * Indicates whether or not this parser is configured to
     * validate XML documents.
     *
     * @return true if this parser is configured to validate
     *         XML documents; false otherwise.
     */

    public abstract boolean isValidating();

    /**
     * Specify the {@link EntityResolver} to be used to resolve
     * entities present in the XML document to be parsed. Setting
     * this to <code>null</code> will result in the underlying
     * implementation using it's own default implementation and
     * behavior.
     *
     * @param er The <code>EntityResolver</code> to be used to resolve entities
     *           present in the XML document to be parsed.
     */

    public abstract void setEntityResolver(EntityResolver er);

    /**
     * Specify the {@link ErrorHandler} to be used by the parser.
     * Setting this to <code>null</code> will result in the underlying
     * implementation using it's own default implementation and
     * behavior.
     *
     * @param eh The <code>ErrorHandler</code> to be used by the parser.
     */

    public abstract void setErrorHandler(ErrorHandler eh);

    /**
     * Obtain a new instance of a DOM {@link Document} object
     * to build a DOM tree with.
     *
     * @return A new instance of a DOM Document object.
     */

```

```
    public abstract Document newDocument();

    /**
     * Obtain an instance of a {@link DOMImplementation} object.
     *
     * @return A new instance of a <code>DOMImplementation</code>.
     */

    public abstract DOMImplementation getDOMImplementation();

    /** <p>Get current state of canonicalization.</p>
     *
     * @return current state canonicalization control
     */
    /*
    public boolean getCanonicalization() {
        return canonicalState;
    }
    */

    /** <p>Get a reference to the the {@link Schema} being used by
     * the XML processor.</p>
     *
     * <p>If no schema is being used, <code>null</code> is returned.</p>
     *
     * @return {@link Schema} being used or <code>null</code>
     *  if none in use
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     */
    public Schema getSchema() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }


    /**
     * <p>Get the XInclude processing mode for this parser.</p>
     *
     * @return
     *      the return value of
     *      the {@link DocumentBuilderFactory#isXIncludeAware()}
     *      when this parser was created from factory.
     *
     * @throws UnsupportedOperationException When implementation does not
```

```
     *   override this method
     *
     * @since 1.5
     *
     * @see DocumentBuilderFactory#setXIncludeAware(boolean)
     */
    public boolean isXIncludeAware() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }
}
```

16.2 DocumentBuilderFactory.java

Secuencia 155: javax/xml/parsers/DocumentBuilderFactory.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

import javax.xml.validation.Schema;

/**
 * Defines a factory API that enables applications to obtain a
 * parser that produces DOM object trees from XML documents.
 *
```

```
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Neeraj.Bajaj@sun.com">Neeraj Bajaj</a>
 *
 * @version $Revision: 1.9 $, $Date: 2010/05/25 16:19:44 $

 */

public abstract class DocumentBuilderFactory {

    private boolean validating = false;
    private boolean namespaceAware = false;
    private boolean whitespace = false;
    private boolean expandEntityRef = true;
    private boolean ignoreComments = false;
    private boolean coalescing = false;

    /**
     * <p>Protected constructor to prevent instantiation.
     * Use {@link #newInstance()}.</p>
     */
    protected DocumentBuilderFactory () {
    }

    /**
     * Obtain a new instance of a
     * <code>DocumentBuilderFactory</code>. This static method creates
     * a new factory instance.
     * This method uses the following ordered lookup procedure to determine
     * the <code>DocumentBuilderFactory</code> implementation class to
     * load:
     * <ul>
     * <li>
     * Use the <code>javax.xml.parsers.DocumentBuilderFactory</code> system
     * property.
     * </li>
     * <li>
     * Use the properties file "lib/jaxp.properties" in the JRE directory.
     * This configuration file is in standard <code>java.util.Properties
     * </code> format and contains the fully qualified name of the
     * implementation class with the key being the system property defined
     * above.
     *
     * The jaxp.properties file is read only once by the JAXP implementation
     * and it's values are then cached for future use.  If the file does not exist
     * when the first attempt is made to read from it, no further attempts are
     * made to check for its existence.  It is not possible to change the value
     * of any property in jaxp.properties after it has been read for the first time.
     * </li>
     * <li>
     * Uses the service-provider loading facilities, defined by the
     * {@link java.util.ServiceLoader} class, to attempt to locate and load an
     * implementation of the service using the {@linkplain
     * java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
```

```
     * the service-provider loading facility will use the {@linkplain
     * java.lang.Thread#getContextClassLoader() current thread's context class loader}
     * to attempt to load the service. If the context class
     * loader is null, the {@linkplain
     * ClassLoader#getSystemClassLoader() system class loader} will be used.
     * </li>
     * <li>
     * Otherwise, the system-default implementation is returned.
     * </li>
     * </ul>
     *
     * Once an application has obtained a reference to a
     * <code>DocumentBuilderFactory</code> it can use the factory to
     * configure and obtain parser instances.
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems loading {@link DocumentBuilder}s, try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @return New instance of a <code>DocumentBuilderFactory</code>
     *
     * @throws FactoryConfigurationError in case of {@linkplain
     * java.util.ServiceConfigurationError service configuration error} or if
     * the implementation is not available or cannot be instantiated.
     */
    public static DocumentBuilderFactory newInstance() {
        return FactoryFinder.find(
                /* The default property name according to the JAXP spec */
                DocumentBuilderFactory.class, // "javax.xml.parsers.DocumentBuilderFactory"
                /* The fallback implementation class name */
                "com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl");
    }

    /**
     * <p>Obtain a new instance of a <code>DocumentBuilderFactory</code> from class name.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
     * should be loaded.</p>
     *
     * <p>Once an application has obtained a reference to a <code>DocumentBuilderFactory</code>
     * it can use the factory to configure and obtain parser instances.</p>
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
```

```
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <code>javax.xml.parsers.DocumentBuilderFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     *                    current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @return New instance of a <code>DocumentBuilderFactory</code>
     *
     * @throws FactoryConfigurationError if <code>factoryClassName</code> is <code>null</code>, or
     *                                   the factory class cannot be loaded, instantiated.
     *
     * @see #newInstance()
     *
     * @since 1.6
     */
    public static DocumentBuilderFactory newInstance(String factoryClassName, ClassLoader classLoader){
        //do not fallback if given classloader can't find the class, throw exception
        return FactoryFinder.newInstance(DocumentBuilderFactory.class,
                    factoryClassName, classLoader, false);
    }

    /**
     * Creates a new instance of a {@link javax.xml.parsers.DocumentBuilder}
     * using the currently configured parameters.
     *
     * @return A new instance of a DocumentBuilder.
     *
     * @throws ParserConfigurationException if a DocumentBuilder
     *   cannot be created which satisfies the configuration requested.
     */

    public abstract DocumentBuilder newDocumentBuilder()
        throws ParserConfigurationException;


    /**
     * Specifies that the parser produced by this code will
     * provide support for XML namespaces. By default the value of this is set
     * to <code>false</code>
     *
     * @param awareness true if the parser produced will provide support
     *                  for XML namespaces; false otherwise.
     */
```

```java

    public void setNamespaceAware(boolean awareness) {
        this.namespaceAware = awareness;
    }

    /**
     * Specifies that the parser produced by this code will
     * validate documents as they are parsed. By default the value of this
     * is set to <code>false</code>.
     *
     * <p>
     * Note that "the validation" here means
     * <a href="http://www.w3.org/TR/REC-xml#proc-types">a validating
     * parser</a> as defined in the XML recommendation.
     * In other words, it essentially just controls the DTD validation.
     * (except the legacy two properties defined in JAXP 1.2.)
     * </p>
     *
     * <p>
     * To use modern schema languages such as W3C XML Schema or
     * RELAX NG instead of DTD, you can configure your parser to be
     * a non-validating parser by leaving the {@link #setValidating(boolean)}
     * method <code>false</code>, then use the {@link #setSchema(Schema)}
     * method to associate a schema to a parser.
     * </p>
     *
     * @param validating true if the parser produced will validate documents
     *                   as they are parsed; false otherwise.
     */

    public void setValidating(boolean validating) {
        this.validating = validating;
    }

    /**
     * Specifies that the parsers created by this  factory must eliminate
     * whitespace in element content (sometimes known loosely as
     * 'ignorable whitespace') when parsing XML documents (see XML Rec
     * 2.10). Note that only whitespace which is directly contained within
     * element content that has an element only content model (see XML
     * Rec 3.2.1) will be eliminated. Due to reliance on the content model
     * this setting requires the parser to be in validating mode. By default
     * the value of this is set to <code>false</code>.
     *
     * @param whitespace true if the parser created must eliminate whitespace
     *                   in the element content when parsing XML documents;
     *                   false otherwise.
     */

    public void setIgnoringElementContentWhitespace(boolean whitespace) {
        this.whitespace = whitespace;
    }

```

```
    /**
     * Specifies that the parser produced by this code will
     * expand entity reference nodes. By default the value of this is set to
     * <code>true</code>
     *
     * @param expandEntityRef true if the parser produced will expand entity
     *                        reference nodes; false otherwise.
     */

    public void setExpandEntityReferences(boolean expandEntityRef) {
        this.expandEntityRef = expandEntityRef;
    }

    /**
     * <p>Specifies that the parser produced by this code will
     * ignore comments. By default the value of this is set to <code>false
     * </code>.</p>
     *
     * @param ignoreComments <code>boolean</code> value to ignore comments during processing
     */

    public void setIgnoringComments(boolean ignoreComments) {
        this.ignoreComments = ignoreComments;
    }

    /**
     * Specifies that the parser produced by this code will
     * convert CDATA nodes to Text nodes and append it to the
     * adjacent (if any) text node. By default the value of this is set to
     * <code>false</code>
     *
     * @param coalescing  true if the parser produced will convert CDATA nodes
     *                    to Text nodes and append it to the adjacent (if any)
     *                    text node; false otherwise.
     */

    public void setCoalescing(boolean coalescing) {
        this.coalescing = coalescing;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which are namespace aware.
     *
     * @return  true if the factory is configured to produce parsers which
     *          are namespace aware; false otherwise.
     */

    public boolean isNamespaceAware() {
        return namespaceAware;
    }

    /**
```

```
     * Indicates whether or not the factory is configured to produce
     * parsers which validate the XML content during parse.
     *
     * @return  true if the factory is configured to produce parsers
     *          which validate the XML content during parse; false otherwise.
     */

    public boolean isValidating() {
        return validating;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which ignore ignorable whitespace in element content.
     *
     * @return  true if the factory is configured to produce parsers
     *          which ignore ignorable whitespace in element content;
     *          false otherwise.
     */

    public boolean isIgnoringElementContentWhitespace() {
        return whitespace;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which expand entity reference nodes.
     *
     * @return  true if the factory is configured to produce parsers
     *          which expand entity reference nodes; false otherwise.
     */

    public boolean isExpandEntityReferences() {
        return expandEntityRef;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which ignores comments.
     *
     * @return  true if the factory is configured to produce parsers
     *          which ignores comments; false otherwise.
     */

    public boolean isIgnoringComments() {
        return ignoreComments;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which converts CDATA nodes to Text nodes and appends it to
     * the adjacent (if any) Text node.
     *
```

```
     * @return  true if the factory is configured to produce parsers
     *          which converts CDATA nodes to Text nodes and appends it to
     *          the adjacent (if any) Text node; false otherwise.
     */

    public boolean isCoalescing() {
        return coalescing;
    }

    /**
     * Allows the user to set specific attributes on the underlying
     * implementation.
     * <p>
     * All implementations that implement JAXP 1.5 or newer are required to
     * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} and
     * {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} properties.
     * </p>
     * <ul>
     *   <li>
     *      <p>
     *      Setting the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property
     *      restricts the access to external DTDs, external Entity References to the
     *      protocols specified by the property.
     *      If access is denied during parsing due to the restriction of this property,
     *      {@link org.xml.sax.SAXException} will be thrown by the parse methods defined by
     *      {@link javax.xml.parsers.DocumentBuilder}.
     *      </p>
     *      <p>
     *      Setting the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} property
     *      restricts the access to external Schema set by the schemaLocation attribute to
     *      the protocols specified by the property.  If access is denied during parsing
     *      due to the restriction of this property, {@link org.xml.sax.SAXException}
     *      will be thrown by the parse methods defined by
     *      {@link javax.xml.parsers.DocumentBuilder}.
     *      </p>
     *   </li>
     * </ul>
     *
     * @param name The name of the attribute.
     * @param value The value of the attribute.
     *
     * @throws IllegalArgumentException thrown if the underlying
     *   implementation doesn't recognize the attribute.
     */
    public abstract void setAttribute(String name, Object value)
                throws IllegalArgumentException;

    /**
     * Allows the user to retrieve specific attributes on the underlying
     * implementation.
     *
     * @param name The name of the attribute.
     *
```

```
     * @return value The value of the attribute.
     *
     * @throws IllegalArgumentException thrown if the underlying
     *   implementation doesn't recognize the attribute.
     */
    public abstract Object getAttribute(String name)
                throws IllegalArgumentException;

    /**
     * <p>Set a feature for this <code>DocumentBuilderFactory</code> and <code>DocumentBuilder</
         code>s created by this factory.</p>
     *
     * <p>
     * Feature names are fully qualified {@link java.net.URI}s.
     * Implementations may define their own features.
     * A {@link ParserConfigurationException} is thrown if this <code>DocumentBuilderFactory</code>
          or the
     * <code>DocumentBuilder</code>s it creates cannot support the feature.
     * It is possible for a <code>DocumentBuilderFactory</code> to expose a feature value but be
         unable to change its state.
     * </p>
     *
     * <p>
     * All implementations are required to support the {@link javax.xml.XMLConstants#
         FEATURE_SECURE_PROCESSING} feature.
     * When the feature is:</p>
     * <ul>
     *   <li>
     *     <code>true</code>: the implementation will limit XML processing to conform to
         implementation limits.
     *     Examples include enity expansion limits and XML Schema constructs that would consume
         large amounts of resources.
     *     If XML processing is limited for security reasons, it will be reported via a call to the
          registered
     *    {@link org.xml.sax.ErrorHandler#fatalError(SAXParseException exception)}.
     *     See {@link  DocumentBuilder#setErrorHandler(org.xml.sax.ErrorHandler errorHandler)}.
     *   </li>
     *   <li>
     *     <code>false</code>: the implementation will processing XML according to the XML
         specifications without
     *     regard to possible implementation limits.
     *   </li>
     * </ul>
     *
     * @param name Feature name.
     * @param value Is feature state <code>true</code> or <code>false</code>.
     *
     * @throws ParserConfigurationException if this <code>DocumentBuilderFactory</code> or the <
         code>DocumentBuilder</code>s
     *   it creates cannot support this feature.
     * @throws NullPointerException If the <code>name</code> parameter is null.
     */
    public abstract void setFeature(String name, boolean value)
```

```
            throws ParserConfigurationException;

    /**
     * <p>Get the state of the named feature.</p>
     *
     * <p>
     * Feature names are fully qualified {@link java.net.URI}s.
     * Implementations may define their own features.
     * An {@link ParserConfigurationException} is thrown if this <code>DocumentBuilderFactory</code>
       > or the
     * <code>DocumentBuilder</code>s it creates cannot support the feature.
     * It is possible for an <code>DocumentBuilderFactory</code> to expose a feature value but be
       unable to change its state.
     * </p>
     *
     * @param name Feature name.
     *
     * @return State of the named feature.
     *
     * @throws ParserConfigurationException if this <code>DocumentBuilderFactory</code>
     *   or the <code>DocumentBuilder</code>s it creates cannot support this feature.
     */
    public abstract boolean getFeature(String name)
            throws ParserConfigurationException;


    /**
     * Gets the {@link Schema} object specified through
     * the {@link #setSchema(Schema schema)} method.
     *
     * @return
     *      the {@link Schema} object that was last set through
     *      the {@link #setSchema(Schema)} method, or null
     *      if the method was not invoked since a {@link DocumentBuilderFactory}
     *      is created.
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
     * @since 1.5
     */
    public Schema getSchema() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );

    }

    /**
```

```
     * <p>Set the {@link Schema} to be used by parsers created
     * from this factory.
     *
     * <p>
     * When a {@link Schema} is non-null, a parser will use a validator
     * created from it to validate documents before it passes information
     * down to the application.
     *
     * <p>When errors are found by the validator, the parser is responsible
     * to report them to the user-specified {@link org.xml.sax.ErrorHandler}
     * (or if the error handler is not set, ignore them or throw them), just
     * like any other errors found by the parser itself.
     * In other words, if the user-specified {@link org.xml.sax.ErrorHandler}
     * is set, it must receive those errors, and if not, they must be
     * treated according to the implementation specific
     * default error handling rules.
     *
     * <p>
     * A validator may modify the outcome of a parse (for example by
     * adding default values that were missing in documents), and a parser
     * is responsible to make sure that the application will receive
     * modified DOM trees.
     *
     * <p>
     * Initialy, null is set as the {@link Schema}.
     *
     * <p>
     * This processing will take effect even if
     * the {@link #isValidating()} method returns <code>false</code>.
     *
     * <p>It is an error to use
     * the <code>http://java.sun.com/xml/jaxp/properties/schemaSource</code>
     * property and/or the <code>http://java.sun.com/xml/jaxp/properties/schemaLanguage</code>
     * property in conjunction with a {@link Schema} object.
     * Such configuration will cause a {@link ParserConfigurationException}
     * exception when the {@link #newDocumentBuilder()} is invoked.</p>
     *
     *
     * <h4>Note for implmentors</h4>
     *
     * <p>
     * A parser must be able to work with any {@link Schema}
     * implementation. However, parsers and schemas are allowed
     * to use implementation-specific custom mechanisms
     * as long as they yield the result described in the specification.
     * </p>
     *
     * @param schema <code>Schema</code> to use or <code>null</code>
     *   to remove a schema.
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
```

```
     * @since 1.5
     */
    public void setSchema(Schema schema) {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }



    /**
     * <p>Set state of XInclude processing.</p>
     *
     * <p>If XInclude markup is found in the document instance, should it be
     * processed as specified in <a href="http://www.w3.org/TR/xinclude/">
     * XML Inclusions (XInclude) Version 1.0</a>.</p>
     *
     * <p>XInclude processing defaults to <code>false</code>.</p>
     *
     * @param state Set XInclude processing to <code>true</code> or
     *   <code>false</code>
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
     * @since 1.5
     */
    public void setXIncludeAware(final boolean state) {
        if (state) {
            throw new UnsupportedOperationException(" setXIncludeAware " +
                "is not supported on this JAXP" +
                " implementation or earlier: " + this.getClass());
        }
    }

    /**
     * <p>Get state of XInclude processing.</p>
     *
     * @return current state of XInclude processing
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
     * @since 1.5
     */
    public boolean isXIncludeAware() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
```

```
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }
}
```

16.3 FactoryConfigurationError.java

Secuencia 156: javax/xml/parsers/FactoryConfigurationError.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

/**
 * Thrown when a problem with configuration with the Parser Factories
 * exists. This error will typically be thrown when the class of a
 * parser factory specified in the system properties cannot be found
 * or instantiated.
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @version $Revision: 1.7 $, $Date: 2010-11-01 04:36:09 $
 */

public class FactoryConfigurationError extends Error {
    private static final long serialVersionUID = -827108682472263355L;

    /**
     *<code>Exception</code> that represents the error.
     */
```

```
    private Exception exception;

    /**
     * Create a new <code>FactoryConfigurationError</code> with no
     * detail mesage.
     */

    public FactoryConfigurationError() {
        super();
        this.exception = null;
    }

    /**
     * Create a new <code>FactoryConfigurationError</code> with
     * the <code>String </code> specified as an error message.
     *
     * @param msg The error message for the exception.
     */

    public FactoryConfigurationError(String msg) {
        super(msg);
        this.exception = null;
    }


    /**
     * Create a new <code>FactoryConfigurationError</code> with a
     * given <code>Exception</code> base cause of the error.
     *
     * @param e The exception to be encapsulated in a
     * FactoryConfigurationError.
     */

    public FactoryConfigurationError(Exception e) {
        super(e.toString());
        this.exception = e;
    }

    /**
     * Create a new <code>FactoryConfigurationError</code> with the
     * given <code>Exception</code> base cause and detail message.
     *
     * @param e The exception to be encapsulated in a
     * FactoryConfigurationError
     * @param msg The detail message.
     */

    public FactoryConfigurationError(Exception e, String msg) {
        super(msg);
        this.exception = e;
    }


```

```
    /**
     * Return the message (if any) for this error . If there is no
     * message for the exception and there is an encapsulated
     * exception then the message of that exception, if it exists will be
     * returned. Else the name of the encapsulated exception will be
     * returned.
     *
     * @return The error message.
     */

    public String getMessage () {
        String message = super.getMessage ();

        if (message == null && exception != null) {
            return exception.getMessage();
        }

        return message;
    }

    /**
     * Return the actual exception (if any) that caused this exception to
     * be raised.
     *
     * @return The encapsulated exception, or null if there is none.
     */

    public Exception getException () {
        return exception;
    }

    /**
     * use the exception chaining mechanism of JDK1.4
    */
    @Override
    public Throwable getCause() {
        return exception;
    }
}
```

16.4 FactoryFinder.java

Secuencia 157: javax/xml/parsers/FactoryFinder.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

import java.io.File;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Iterator;
import java.util.Properties;
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * <p>Implements pluggable Parsers.</p>
 *
 * <p>This class is duplicated for each JAXP subpackage so keep it in
 * sync.  It is package private for secure class loading.</p>
 *
 * @author Santiago.PericasGeertsen@sun.com
 * @author Huizhe.Wang@oracle.com
 */
class FactoryFinder {
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xerces.internal";
    /**
     * Internal debug flag.
     */
    private static boolean debug = false;

    /**
     * Cache for properties in java.home/lib/jaxp.properties
     */
    private static final Properties cacheProps = new Properties();

    /**
     * Flag indicating if properties from java.home/lib/jaxp.properties
     * have been cached.
     */
    static volatile boolean firstTime = true;

    /**
```

```
     * Security support class use to check access control before
     * getting certain system resources.
     */
    private static final SecuritySupport ss = new SecuritySupport();

    // Define system property "jaxp.debug" to get output
    static {
        // Use try/catch block to support applets, which throws
        // SecurityException out of this code.
        try {
            String val = ss.getSystemProperty("jaxp.debug");
            // Allow simply setting the prop to turn on debug
            debug = val != null && !"false".equals(val);
        }
        catch (SecurityException se) {
            debug = false;
        }
    }

    private static void dPrint(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * Attempt to load a class using the class loader supplied. If that fails
     * and fall back is enabled, the current (i.e. bootstrap) class loader is
     * tried.
     *
     * If the class loader supplied is <code>null</code>, first try using the
     * context class loader followed by the current (i.e. bootstrap) class
     * loader.
     *
     * Use bootstrap classLoader if cl = null and useBSClsLoader is true
     */
    static private Class<?> getProviderClass(String className, ClassLoader cl,
            boolean doFallback, boolean useBSClsLoader) throws ClassNotFoundException
    {
        try {
            if (cl == null) {
                if (useBSClsLoader) {
                    return Class.forName(className, false, FactoryFinder.class.getClassLoader());
                } else {
                    cl = ss.getContextClassLoader();
                    if (cl == null) {
                        throw new ClassNotFoundException();
                    }
                    else {
                        return Class.forName(className, false, cl);
                    }
                }
            }
```

```
            else {
                return Class.forName(className, false, cl);
            }
        }
        catch (ClassNotFoundException e1) {
            if (doFallback) {
                // Use current class loader - should always be bootstrap CL
                return Class.forName(className, false, FactoryFinder.class.getClassLoader());
            }
            else {
                throw e1;
            }
        }
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl,
                             boolean doFallback)
        throws FactoryConfigurationError
    {
        return newInstance(type, className, cl, doFallback, false);
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
```

```
     *
     * @param useBSClsLoader True if cl=null actually meant bootstrap classLoader. This parameter
     * is needed since DocumentBuilderFactory/SAXParserFactory defined null as context classLoader.
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl,
                             boolean doFallback, boolean useBSClsLoader)
        throws FactoryConfigurationError
    {
        assert type != null;
        // make sure we have access to restricted packages
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                cl = null;
                useBSClsLoader = true;
            }
        }

        try {
            Class<?> providerClass = getProviderClass(className, cl, doFallback, useBSClsLoader);
            if (!type.isAssignableFrom(providerClass)) {
                throw new ClassCastException(className + " cannot be cast to " + type.getName());
            }
            Object instance = providerClass.newInstance();
            if (debug) {    // Extra check to avoid computing cl strings
                dPrint("created new instance of " + providerClass +
                       " using ClassLoader: " + cl);
            }
            return type.cast(instance);
        }
        catch (ClassNotFoundException x) {
            throw new FactoryConfigurationError(x,
                "Provider " + className + " not found");
        }
        catch (Exception x) {
            throw new FactoryConfigurationError(x,
                "Provider " + className + " could not be instantiated: " + x);
        }
    }

    /**
     * Finds the implementation Class object in the specified order.  Main
     * entry point.
     * @return Class object of factory, never null
     *
     * @param type                  Base class / Service interface  of the
     *                              factory to find.
     * @param fallbackClassName     Implementation class name, if nothing else
     *                              is found.  Use null to mean no fallback.
     *
     * Package private so this code can be shared.
     */
    static <T> T find(Class<T> type, String fallbackClassName)
        throws FactoryConfigurationError
```

```
    {
        final String factoryId = type.getName();
        dPrint("find factoryId =" + factoryId);

        // Use the system property first
        try {
            String systemProp = ss.getSystemProperty(factoryId);
            if (systemProp != null) {
                dPrint("found system property, value=" + systemProp);
                return newInstance(type, systemProp, null, true);
            }
        }
        catch (SecurityException se) {
            if (debug) se.printStackTrace();
        }

        // try to read from $java.home/lib/jaxp.properties
        try {
            if (firstTime) {
                synchronized (cacheProps) {
                    if (firstTime) {
                        String configFile = ss.getSystemProperty("java.home") + File.separator +
                            "lib" + File.separator + "jaxp.properties";
                        File f = new File(configFile);
                        firstTime = false;
                        if (ss.doesFileExist(f)) {
                            dPrint("Read properties file "+f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                    }
                }
            }
            final String factoryClassName = cacheProps.getProperty(factoryId);

            if (factoryClassName != null) {
                dPrint("found in $java.home/jaxp.properties, value=" + factoryClassName);
                return newInstance(type, factoryClassName, null, true);
            }
        }
        catch (Exception ex) {
            if (debug) ex.printStackTrace();
        }

        // Try Jar Service Provider Mechanism
        T provider = findServiceProvider(type);
        if (provider != null) {
            return provider;
        }
        if (fallbackClassName == null) {
            throw new FactoryConfigurationError(
                "Provider for " + factoryId + " cannot be found");
        }

```

```
        dPrint("loaded from fallback value: " + fallbackClassName);
        return newInstance(type, fallbackClassName, null, true);
    }

    /*
     * Try to find provider using the ServiceLoader API
     *
     * @param type Base class / Service interface  of the factory to find.
     *
     * @return instance of provider class if found or null
     */
    private static <T> T findServiceProvider(final Class<T> type) {
        try {
            return AccessController.doPrivileged(new PrivilegedAction<T>() {
                public T run() {
                    final ServiceLoader<T> serviceLoader = ServiceLoader.load(type);
                    final Iterator<T> iterator = serviceLoader.iterator();
                    if (iterator.hasNext()) {
                        return iterator.next();
                    } else {
                        return null;
                    }
                }
            });
        } catch(ServiceConfigurationError e) {
            // It is not possible to wrap an error directly in
            // FactoryConfigurationError - so we need to wrap the
            // ServiceConfigurationError in a RuntimeException.
            // The alternative would be to modify the logic in
            // FactoryConfigurationError to allow setting a
            // Throwable as the cause, but that could cause
            // compatibility issues down the road.
            final RuntimeException x = new RuntimeException(
                    "Provider for " + type + " cannot be created", e);
            final FactoryConfigurationError error =
                    new FactoryConfigurationError(x, x.getMessage());
            throw error;
        }
    }

}
```

16.5 ParserConfigurationException.java

Secuencia 158: javax/xml/parsers/ParserConfigurationException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

/**
 * Indicates a serious configuration error.
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */

public class ParserConfigurationException extends Exception {

    /**
     * Create a new <code>ParserConfigurationException</code> with no
     * detail mesage.
     */

    public ParserConfigurationException() {
        super();
    }

    /**
     * Create a new <code>ParserConfigurationException</code> with
     * the <code>String</code> specified as an error message.
     *
     * @param msg The error message for the exception.
     */

    public ParserConfigurationException(String msg) {
        super(msg);
    }

}
```

16.6 SAXParser.java

Secuencia 159: javax/xml/parsers/SAXParser.java

```
/*
```

```
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;

import javax.xml.validation.Schema;

import org.xml.sax.HandlerBase;
import org.xml.sax.InputSource;
import org.xml.sax.Parser;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;


/**
 * Defines the API that wraps an {@link org.xml.sax.XMLReader}
 * implementation class. In JAXP 1.0, this class wrapped the
 * {@link org.xml.sax.Parser} interface, however this interface was
 * replaced by the {@link org.xml.sax.XMLReader}. For ease
 * of transition, this class continues to support the same name
 * and interface as well as supporting new methods.
 *
 * An instance of this class can be obtained from the
 * {@link javax.xml.parsers.SAXParserFactory#newSAXParser()} method.
 * Once an instance of this class is obtained, XML can be parsed from
```

```
 * a variety of input sources. These input sources are InputStreams,
 * Files, URLs, and SAX InputSources.<p>
 *
 * This static method creates a new factory instance based
 * on a system property setting or uses the platform default
 * if no property has been defined.<p>
 *
 * The system property that controls which Factory implementation
 * to create is named <code>&quot;javax.xml.parsers.SAXParserFactory&quot;</code>.
 * This property names a class that is a concrete subclass of this
 * abstract class. If no property is defined, a platform default
 * will be used.</p>
 *
 * As the content is parsed by the underlying parser, methods of the
 * given {@link org.xml.sax.HandlerBase} or the
 * {@link org.xml.sax.helpers.DefaultHandler} are called.<p>
 *
 * Implementors of this class which wrap an underlaying implementation
 * can consider using the {@link org.xml.sax.helpers.ParserAdapter}
 * class to initially adapt their SAX1 implementation to work under
 * this revised class.
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public abstract class SAXParser {

    /**
     * <p>Protected constructor to prevent instaniation.
     * Use {@link javax.xml.parsers.SAXParserFactory#newSAXParser()}.</p>
     */
    protected SAXParser () {

    }

        /**
         * <p>Reset this <code>SAXParser</code> to its original configuration.</p>
         *
         * <p><code>SAXParser</code> is reset to the same state as when it was created with
         * {@link SAXParserFactory#newSAXParser()}.
         * <code>reset()</code> is designed to allow the reuse of existing <code>SAXParser</code>s
         * thus saving resources associated with the creation of new <code>SAXParser</code>s.</p>
         *
         * <p>The reset <code>SAXParser</code> is not guaranteed to have the same {@link Schema}
         * <code>Object</code>, e.g. {@link Object#equals(Object obj)}.  It is guaranteed to have a functionally equal
         * <code>Schema</code>.</p>
     *
     * @throws UnsupportedOperationException When Implementations do not
     *   override this method
         *
         * @since 1.5
         */
        public void reset() {
```

```
                // implementors should override this method
                throw new UnsupportedOperationException(
                        "This SAXParser, \"" + this.getClass().getName() + "\", does not support the reset functionality."
                        + "  Specification \"" + this.getClass().getPackage().getSpecificationTitle() + "\""
                        + " version \"" + this.getClass().getPackage().getSpecificationVersion() + "\""
                        );
        }

    /**
     * <p>Parse the content of the given {@link java.io.InputStream}
     * instance as XML using the specified {@link org.xml.sax.HandlerBase}.
     * <i> Use of the DefaultHandler version of this method is recommended as
     * the HandlerBase class has been deprecated in SAX 2.0</i>.</p>
     *
     * @param is InputStream containing the content to be parsed.
     * @param hb The SAX HandlerBase to use.
     *
     * @throws IllegalArgumentException If the given InputStream is null.
     * @throws SAXException If parse produces a SAX error.
     * @throws IOException If an IO error occurs interacting with the
     *   <code>InputStream</code>.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(InputStream is, HandlerBase hb)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource input = new InputSource(is);
        this.parse(input, hb);
    }

    /**
     * <p>Parse the content of the given {@link java.io.InputStream}
     * instance as XML using the specified {@link org.xml.sax.HandlerBase}.
     * <i> Use of the DefaultHandler version of this method is recommended as
     * the HandlerBase class has been deprecated in SAX 2.0</i>.</p>
     *
     * @param is InputStream containing the content to be parsed.
     * @param hb The SAX HandlerBase to use.
     * @param systemId The systemId which is needed for resolving relative URIs.
     *
     * @throws IllegalArgumentException If the given <code>InputStream</code> is
     *   <code>null</code>.
     * @throws IOException If any IO error occurs interacting with the
     *   <code>InputStream</code>.
     * @throws SAXException If any SAX errors occur during processing.
```

```
     *
     * @see org.xml.sax.DocumentHandler version of this method instead.
     */
    public void parse(
        InputStream is,
        HandlerBase hb,
        String systemId)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource input = new InputSource(is);
        input.setSystemId(systemId);
        this.parse(input, hb);
    }

    /**
     * Parse the content of the given {@link java.io.InputStream}
     * instance as XML using the specified
     * {@link org.xml.sax.helpers.DefaultHandler}.
     *
     * @param is InputStream containing the content to be parsed.
     * @param dh The SAX DefaultHandler to use.
     *
     * @throws IllegalArgumentException If the given InputStream is null.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(InputStream is, DefaultHandler dh)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource input = new InputSource(is);
        this.parse(input, dh);
    }

    /**
     * Parse the content of the given {@link java.io.InputStream}
     * instance as XML using the specified
     * {@link org.xml.sax.helpers.DefaultHandler}.
     *
     * @param is InputStream containing the content to be parsed.
     * @param dh The SAX DefaultHandler to use.
     * @param systemId The systemId which is needed for resolving relative URIs.
     *
     * @throws IllegalArgumentException If the given InputStream is null.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
```

```
     *
     * @see org.xml.sax.DocumentHandler version of this method instead.
     */
    public void parse(
        InputStream is,
        DefaultHandler dh,
        String systemId)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputStream cannot be null");
        }

        InputSource input = new InputSource(is);
        input.setSystemId(systemId);
        this.parse(input, dh);
    }

    /**
     * Parse the content described by the giving Uniform Resource
     * Identifier (URI) as XML using the specified
     * {@link org.xml.sax.HandlerBase}.
     * <i> Use of the DefaultHandler version of this method is recommended as
     * the <code>HandlerBase</code> class has been deprecated in SAX 2.0</i>
     *
     * @param uri The location of the content to be parsed.
     * @param hb The SAX HandlerBase to use.
     *
     * @throws IllegalArgumentException If the uri is null.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(String uri, HandlerBase hb)
        throws SAXException, IOException {
        if (uri == null) {
            throw new IllegalArgumentException("uri cannot be null");
        }

        InputSource input = new InputSource(uri);
        this.parse(input, hb);
    }

    /**
     * Parse the content described by the giving Uniform Resource
     * Identifier (URI) as XML using the specified
     * {@link org.xml.sax.helpers.DefaultHandler}.
     *
     * @param uri The location of the content to be parsed.
     * @param dh The SAX DefaultHandler to use.
     *
     * @throws IllegalArgumentException If the uri is null.
     * @throws IOException If any IO errors occur.
```

```
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(String uri, DefaultHandler dh)
        throws SAXException, IOException {
        if (uri == null) {
            throw new IllegalArgumentException("uri cannot be null");
        }

        InputSource input = new InputSource(uri);
        this.parse(input, dh);
    }

    /**
     * Parse the content of the file specified as XML using the
     * specified {@link org.xml.sax.HandlerBase}.
     * <i> Use of the DefaultHandler version of this method is recommended as
     * the HandlerBase class has been deprecated in SAX 2.0</i>
     *
     * @param f The file containing the XML to parse
     * @param hb The SAX HandlerBase to use.
     *
     * @throws IllegalArgumentException If the File object is null.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(File f, HandlerBase hb)
        throws SAXException, IOException {
        if (f == null) {
            throw new IllegalArgumentException("File cannot be null");
        }

        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        InputSource input = new InputSource(f.toURI().toASCIIString());
        this.parse(input, hb);
    }

    /**
     * Parse the content of the file specified as XML using the
     * specified {@link org.xml.sax.helpers.DefaultHandler}.
     *
     * @param f The file containing the XML to parse
     * @param dh The SAX DefaultHandler to use.
     *
     * @throws IllegalArgumentException If the File object is null.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
```

```
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(File f, DefaultHandler dh)
        throws SAXException, IOException {
        if (f == null) {
            throw new IllegalArgumentException("File cannot be null");
        }

        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        InputSource input = new InputSource(f.toURI().toASCIIString());
        this.parse(input, dh);
    }

    /**
     * Parse the content given {@link org.xml.sax.InputSource}
     * as XML using the specified
     * {@link org.xml.sax.HandlerBase}.
     * <i> Use of the DefaultHandler version of this method is recommended as
     * the HandlerBase class has been deprecated in SAX 2.0</i>
     *
     * @param is The InputSource containing the content to be parsed.
     * @param hb The SAX HandlerBase to use.
     *
     * @throws IllegalArgumentException If the <code>InputSource</code> object
     *   is <code>null</code>.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(InputSource is, HandlerBase hb)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputSource cannot be null");
        }

        Parser parser = this.getParser();
        if (hb != null) {
            parser.setDocumentHandler(hb);
            parser.setEntityResolver(hb);
            parser.setErrorHandler(hb);
            parser.setDTDHandler(hb);
        }
        parser.parse(is);
    }

    /**
     * Parse the content given {@link org.xml.sax.InputSource}
     * as XML using the specified
     * {@link org.xml.sax.helpers.DefaultHandler}.
     *
```

```
     * @param is The InputSource containing the content to be parsed.
     * @param dh The SAX DefaultHandler to use.
     *
     * @throws IllegalArgumentException If the <code>InputSource</code> object
     *   is <code>null</code>.
     * @throws IOException If any IO errors occur.
     * @throws SAXException If any SAX errors occur during processing.
     *
     * @see org.xml.sax.DocumentHandler
     */
    public void parse(InputSource is, DefaultHandler dh)
        throws SAXException, IOException {
        if (is == null) {
            throw new IllegalArgumentException("InputSource cannot be null");
        }

        XMLReader reader = this.getXMLReader();
        if (dh != null) {
            reader.setContentHandler(dh);
            reader.setEntityResolver(dh);
            reader.setErrorHandler(dh);
            reader.setDTDHandler(dh);
        }
        reader.parse(is);
    }

    /**
     * Returns the SAX parser that is encapsultated by the
     * implementation of this class.
     *
     * @return The SAX parser that is encapsultated by the
     *         implementation of this class.
     *
     * @throws SAXException If any SAX errors occur during processing.
     */
    public abstract org.xml.sax.Parser getParser() throws SAXException;

    /**
     * Returns the {@link org.xml.sax.XMLReader} that is encapsulated by the
     * implementation of this class.
     *
     * @return The XMLReader that is encapsulated by the
     *         implementation of this class.
     *
     * @throws SAXException If any SAX errors occur during processing.
     */

    public abstract org.xml.sax.XMLReader getXMLReader() throws SAXException;

    /**
     * Indicates whether or not this parser is configured to
     * understand namespaces.
     *
```

```
     * @return true if this parser is configured to
     *         understand namespaces; false otherwise.
     */

    public abstract boolean isNamespaceAware();

    /**
     * Indicates whether or not this parser is configured to
     * validate XML documents.
     *
     * @return true if this parser is configured to
     *         validate XML documents; false otherwise.
     */

    public abstract boolean isValidating();

    /**
     * <p>Sets the particular property in the underlying implementation of
     * {@link org.xml.sax.XMLReader}.
     * A list of the core features and properties can be found at
     * <a href="http://sax.sourceforge.net/?selected=get-set">
     * http://sax.sourceforge.net/?selected=get-set</a>.</p>
     * <p>
     * All implementations that implement JAXP 1.5 or newer are required to
     * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} and
     * {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} properties.
     * </p>
     * <ul>
     *   <li>
     *      <p>
     *      Setting the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property
     *      restricts the access to external DTDs, external Entity References to
     *      the protocols specified by the property.  If access is denied during parsing
     *      due to the restriction of this property, {@link org.xml.sax.SAXException}
     *      will be thrown by the parse methods defined by {@link javax.xml.parsers.SAXParser}.
     *      </p>
     *      <p>
     *      Setting the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} property
     *      restricts the access to external Schema set by the schemaLocation attribute to
     *      the protocols specified by the property.  If access is denied during parsing
     *      due to the restriction of this property, {@link org.xml.sax.SAXException}
     *      will be thrown by the parse methods defined by the {@link javax.xml.parsers.SAXParser}.
     *      </p>
     *   </li>
     * </ul>
     *
     * @param name The name of the property to be set.
     * @param value The value of the property to be set.
     *
     * @throws SAXNotRecognizedException When the underlying XMLReader does
     *   not recognize the property name.
     * @throws SAXNotSupportedException When the underlying XMLReader
     *  recognizes the property name but doesn't support the property.
```

```
     *
     * @see org.xml.sax.XMLReader#setProperty
     */
    public abstract void setProperty(String name, Object value)
        throws SAXNotRecognizedException, SAXNotSupportedException;

    /**
     * <p>Returns the particular property requested for in the underlying
     * implementation of {@link org.xml.sax.XMLReader}.</p>
     *
     * @param name The name of the property to be retrieved.
     * @return Value of the requested property.
     *
     * @throws SAXNotRecognizedException When the underlying XMLReader does
     *    not recognize the property name.
     * @throws SAXNotSupportedException When the underlying XMLReader
     *  recognizes the property name but doesn't support the property.
     *
     * @see org.xml.sax.XMLReader#getProperty
     */
    public abstract Object getProperty(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException;

    /** <p>Get current state of canonicalization.</p>
     *
     * @return current state canonicalization control
     */
    /*
    public boolean getCanonicalization() {
        return canonicalState;
    }
    */

    /** <p>Get a reference to the the {@link Schema} being used by
     * the XML processor.</p>
     *
     * <p>If no schema is being used, <code>null</code> is returned.</p>
     *
     * @return {@link Schema} being used or <code>null</code>
     *  if none in use
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     */
    public Schema getSchema() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
```

```
            );
    }

    /**
     * <p>Get the XInclude processing mode for this parser.</p>
     *
     * @return
     *      the return value of
     *      the {@link SAXParserFactory#isXIncludeAware()}
     *      when this parser was created from factory.
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     *
     * @see SAXParserFactory#setXIncludeAware(boolean)
     */
    public boolean isXIncludeAware() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }
}
```

16.7 SAXParserFactory.java

Secuencia 160: javax/xml/parsers/SAXParserFactory.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

package javax.xml.parsers;

import javax.xml.validation.Schema;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;

/**
 * Defines a factory API that enables applications to configure and
 * obtain a SAX based parser to parse XML documents.
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Neeraj.Bajaj@sun.com">Neeraj Bajaj</a>
 *
 * @version $Revision: 1.9 $, $Date: 2010/05/25 16:19:44 $
 *
 */
public abstract class SAXParserFactory {

    /**
     * <p>Should Parsers be validating?</p>
     */
    private boolean validating = false;

    /**
     * <p>Should Parsers be namespace aware?</p>
     */
    private boolean namespaceAware = false;

    /**
     * <p>Protected constructor to force use of {@link #newInstance()}.</p>
     */
    protected SAXParserFactory () {

    }

    /**
     * Obtain a new instance of a <code>SAXParserFactory</code>. This
     * static method creates a new factory instance
     * This method uses the following ordered lookup procedure to determine
     * the <code>SAXParserFactory</code> implementation class to
     * load:
     * <ul>
     * <li>
     * Use the <code>javax.xml.parsers.SAXParserFactory</code> system
     * property.
     * </li>
     * <li>
     * Use the properties file "lib/jaxp.properties" in the JRE directory.
```

```
     * This configuration file is in standard <code>java.util.Properties
     * </code> format and contains the fully qualified name of the
     * implementation class with the key being the system property defined
     * above.
     *
     * The jaxp.properties file is read only once by the JAXP implementation
     * and it's values are then cached for future use.  If the file does not exist
     * when the first attempt is made to read from it, no further attempts are
     * made to check for its existence.  It is not possible to change the value
     * of any property in jaxp.properties after it has been read for the first time.
     * </li>
     * <li>
     * Use the service-provider loading facilities, defined by the
     * {@link java.util.ServiceLoader} class, to attempt to locate and load an
     * implementation of the service using the {@linkplain
     * java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
     * the service-provider loading facility will use the {@linkplain
     * java.lang.Thread#getContextClassLoader() current thread's context class loader}
     * to attempt to load the service. If the context class
     * loader is null, the {@linkplain
     * ClassLoader#getSystemClassLoader() system class loader} will be used.
     * </li>
     * <li>
     * Otherwise the system-default implementation is returned.
     * </li>
     * </ul>
     *
     * Once an application has obtained a reference to a
     * <code>SAXParserFactory</code> it can use the factory to
     * configure and obtain parser instances.
     *
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems loading {@link SAXParser}s, try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     *
     * @return A new instance of a SAXParserFactory.
     *
     * @throws FactoryConfigurationError in case of {@linkplain
     * java.util.ServiceConfigurationError service configuration error} or if
     * the implementation is not available or cannot be instantiated.
     */

    public static SAXParserFactory newInstance() {
        return FactoryFinder.find(
```

```
                /* The default property name according to the JAXP spec */
                SAXParserFactory.class,
                /* The fallback implementation class name */
                "com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl");
    }

    /**
     * <p>Obtain a new instance of a <code>SAXParserFactory</code> from class name.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
     * should be loaded.</p>
     *
     * <p>Once an application has obtained a reference to a <code>SAXParserFactory</code>
     * it can use the factory to configure and obtain parser instances.</p>
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems, try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <
         code>javax.xml.parsers.SAXParserFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</
         code>
     *                    current <code>Thread</code>'s context classLoader is used to load the
         factory class.
     *
     * @return New instance of a <code>SAXParserFactory</code>
     *
     * @throws FactoryConfigurationError if <code>factoryClassName</code> is <code>null</code>, or
     *                                   the factory class cannot be loaded, instantiated.
     *
     * @see #newInstance()
     *
     * @since 1.6
     */
    public static SAXParserFactory newInstance(String factoryClassName, ClassLoader classLoader){
            //do not fallback if given classloader can't find the class, throw exception
            return FactoryFinder.newInstance(SAXParserFactory.class,
                    factoryClassName, classLoader, false);
    }

    /**
     * <p>Creates a new instance of a SAXParser using the currently
     * configured factory parameters.</p>
     *
```

```
     * @return A new instance of a SAXParser.
     *
     * @throws ParserConfigurationException if a parser cannot
     *   be created which satisfies the requested configuration.
     * @throws SAXException for SAX errors.
     */

    public abstract SAXParser newSAXParser()
        throws ParserConfigurationException, SAXException;


    /**
     * Specifies that the parser produced by this code will
     * provide support for XML namespaces. By default the value of this is set
     * to <code>false</code>.
     *
     * @param awareness true if the parser produced by this code will
     *                  provide support for XML namespaces; false otherwise.
     */

    public void setNamespaceAware(boolean awareness) {
        this.namespaceAware = awareness;
    }

    /**
     * Specifies that the parser produced by this code will
     * validate documents as they are parsed. By default the value of this is
     * set to <code>false</code>.
     *
     * <p>
     * Note that "the validation" here means
     * <a href="http://www.w3.org/TR/REC-xml#proc-types">a validating
     * parser</a> as defined in the XML recommendation.
     * In other words, it essentially just controls the DTD validation.
     * (except the legacy two properties defined in JAXP 1.2.)
     * </p>
     *
     * <p>
     * To use modern schema languages such as W3C XML Schema or
     * RELAX NG instead of DTD, you can configure your parser to be
     * a non-validating parser by leaving the {@link #setValidating(boolean)}
     * method <code>false</code>, then use the {@link #setSchema(Schema)}
     * method to associate a schema to a parser.
     * </p>
     *
     * @param validating true if the parser produced by this code will
     *                   validate documents as they are parsed; false otherwise.
     */

    public void setValidating(boolean validating) {
        this.validating = validating;
    }

```

```
    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which are namespace aware.
     *
     * @return true if the factory is configured to produce
     *         parsers which are namespace aware; false otherwise.
     */

    public boolean isNamespaceAware() {
        return namespaceAware;
    }

    /**
     * Indicates whether or not the factory is configured to produce
     * parsers which validate the XML content during parse.
     *
     * @return true if the factory is configured to produce parsers which validate
     *         the XML content during parse; false otherwise.
     */

    public boolean isValidating() {
        return validating;
    }

    /**
     *
     * <p>Sets the particular feature in the underlying implementation of
     * org.xml.sax.XMLReader.
     * A list of the core features and properties can be found at
     * <a href="http://www.saxproject.org/">http://www.saxproject.org/</a></p>
     *
     * <p>All implementations are required to support the {@link javax.xml.XMLConstants#
         FEATURE_SECURE_PROCESSING} feature.
     * When the feature is</p>
     * <ul>
     *   <li>
     *     <code>true</code>: the implementation will limit XML processing to conform to
         implementation limits.
     *     Examples include entity expansion limits and XML Schema constructs that would consume
         large amounts of resources.
     *     If XML processing is limited for security reasons, it will be reported via a call to the
          registered
     *     {@link org.xml.sax.ErrorHandler#fatalError(SAXParseException exception)}.
     *     See {@link SAXParser} <code>parse</code> methods for handler specification.
     *   </li>
     *   <li>
     *     When the feature is <code>false</code>, the implementation will processing XML according
          to the XML specifications without
     *     regard to possible implementation limits.
     *   </li>
     * </ul>
     *
     * @param name The name of the feature to be set.
```

```
     * @param value The value of the feature to be set.
     *
     * @throws ParserConfigurationException if a parser cannot
     *     be created which satisfies the requested configuration.
     * @throws SAXNotRecognizedException When the underlying XMLReader does
     *            not recognize the property name.
     * @throws SAXNotSupportedException When the underlying XMLReader
     *            recognizes the property name but doesn't support the
     *            property.
     * @throws NullPointerException If the <code>name</code> parameter is null.
     *
     * @see org.xml.sax.XMLReader#setFeature
     */
    public abstract void setFeature(String name, boolean value)
        throws ParserConfigurationException, SAXNotRecognizedException,
                SAXNotSupportedException;

    /**
     *
     * <p>Returns the particular property requested for in the underlying
     * implementation of org.xml.sax.XMLReader.</p>
     *
     * @param name The name of the property to be retrieved.
     *
     * @return Value of the requested property.
     *
     * @throws ParserConfigurationException if a parser cannot be created which satisfies the
         requested configuration.
     * @throws SAXNotRecognizedException When the underlying XMLReader does not recognize the
         property name.
     * @throws SAXNotSupportedException When the underlying XMLReader recognizes the property name
         but doesn't support the property.
     *
     * @see org.xml.sax.XMLReader#getProperty
     */
    public abstract boolean getFeature(String name)
        throws ParserConfigurationException, SAXNotRecognizedException,
                SAXNotSupportedException;


    /**
     * Gets the {@link Schema} object specified through
     * the {@link #setSchema(Schema schema)} method.
     *
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @return
     *      the {@link Schema} object that was last set through
     *      the {@link #setSchema(Schema)} method, or null
     *      if the method was not invoked since a {@link SAXParserFactory}
     *      is created.
```

```
     *
     * @since 1.5
     */
    public Schema getSchema() {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }

    /**
     * <p>Set the {@link Schema} to be used by parsers created
     * from this factory.</p>
     *
     * <p>When a {@link Schema} is non-null, a parser will use a validator
     * created from it to validate documents before it passes information
     * down to the application.</p>
     *
     * <p>When warnings/errors/fatal errors are found by the validator, the parser must
     * handle them as if those errors were found by the parser itself.
     * In other words, if the user-specified {@link org.xml.sax.ErrorHandler}
     * is set, it must receive those errors, and if not, they must be
     * treated according to the implementation specific
     * default error handling rules.
     *
     * <p>A validator may modify the SAX event stream (for example by
     * adding default values that were missing in documents), and a parser
     * is responsible to make sure that the application will receive
     * those modified event stream.</p>
     *
     * <p>Initialy, <code>null</code> is set as the {@link Schema}.</p>
     *
     * <p>This processing will take effect even if
     * the {@link #isValidating()} method returns <code>false</code>.
     *
     * <p>It is an error to use
     * the <code>http://java.sun.com/xml/jaxp/properties/schemaSource</code>
     * property and/or the <code>http://java.sun.com/xml/jaxp/properties/schemaLanguage</code>
     * property in conjunction with a non-null {@link Schema} object.
     * Such configuration will cause a {@link SAXException}
     * exception when those properties are set on a {@link SAXParser}.</p>
     *
     * <h4>Note for implementors</h4>
     * <p>
     * A parser must be able to work with any {@link Schema}
     * implementation. However, parsers and schemas are allowed
     * to use implementation-specific custom mechanisms
     * as long as they yield the result described in the specification.
     * </p>
     *
```

```
     * @param schema <code>Schema</code> to use, <code>null</code> to remove a schema.
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     */
    public void setSchema(Schema schema) {
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }

    /**
     * <p>Set state of XInclude processing.</p>
     *
     * <p>If XInclude markup is found in the document instance, should it be
     * processed as specified in <a href="http://www.w3.org/TR/xinclude/">
     * XML Inclusions (XInclude) Version 1.0</a>.</p>
     *
     * <p>XInclude processing defaults to <code>false</code>.</p>
     *
     * @param state Set XInclude processing to <code>true</code> or
     *   <code>false</code>
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     */
    public void setXIncludeAware(final boolean state) {
        if (state) {
            throw new UnsupportedOperationException(" setXIncludeAware " +
                "is not supported on this JAXP"  +
                " implementation or earlier: " + this.getClass());
        }
    }

    /**
     * <p>Get state of XInclude processing.</p>
     *
     * @return current state of XInclude processing
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method
     *
     * @since 1.5
     */
    public boolean isXIncludeAware() {
```

```
        throw new UnsupportedOperationException(
            "This parser does not support specification \""
            + this.getClass().getPackage().getSpecificationTitle()
            + "\" version \""
            + this.getClass().getPackage().getSpecificationVersion()
            + "\""
            );
    }
}
```

16.8 SecuritySupport.java

Secuencia 161: javax/xml/parsers/SecuritySupport.java

```
/*
 * Copyright (c) 2004, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.parsers;

import java.security.*;
import java.net.*;
import java.io.*;
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {
```

```


    ClassLoader getContextClassLoader() throws SecurityException{
        return (ClassLoader)
                AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                //try {
                cl = Thread.currentThread().getContextClassLoader();
                //} catch (SecurityException ex) { }

                if (cl == null)
                    cl = ClassLoader.getSystemClassLoader();

                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                        final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
```

```
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

}
```

17 Xml Soap (javax.xml.soap package)

17.1 AttachmentPart.java

Secuencia 162: javax/xml/soap/AttachmentPart.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.io.InputStream;
import java.io.Reader;
import java.util.Iterator;
```

```

import javax.activation.DataHandler;

/**
 * A single attachment to a <code>SOAPMessage</code> object. A <code>SOAPMessage</code>
 * object may contain zero, one, or many <code>AttachmentPart</code> objects.
 * Each <code>AttachmentPart</code> object consists of two parts,
 * application-specific content and associated MIME headers. The
 * MIME headers consists of name/value pairs that can be used to
 * identify and describe the content.
 * <p>
 * An <code>AttachmentPart</code> object must conform to certain standards.
 * <OL>
 * <LI>It must conform to <a href="http://www.ietf.org/rfc/rfc2045.txt">
 *     MIME [RFC2045] standards</a>
 * <LI>It MUST contain content
 * <LI>The header portion MUST include the following header:
 *  <UL>
 *   <LI><code>Content-Type</code><br>
 *       This header identifies the type of data in the content of an
 *       <code>AttachmentPart</code> object and MUST conform to [RFC2045].
 *       The following is an example of a Content-Type header:
 *       <PRE>
 *       Content-Type:  application/xml
 *       </PRE>
 *       The following line of code, in which <code>ap</code> is an
 *       <code>AttachmentPart</code> object, sets the header shown in
 *       the previous example.
 *       <PRE>
 *       ap.setMimeHeader("Content-Type", "application/xml");
 *       </PRE>
 * <p>
 *  </UL>
 * </OL>
 * <p>
 * There are no restrictions on the content portion of an <code>
 * AttachmentPart</code> object. The content may be anything from a
 * simple plain text object to a complex XML document or image file.
 *
 * <p>
 * An <code>AttachmentPart</code> object is created with the method
 * <code>SOAPMessage.createAttachmentPart</code>. After setting its MIME headers,
 *  the <code>AttachmentPart</code> object is added to the message
 * that created it with the method <code>SOAPMessage.addAttachmentPart</code>.
 *
 * <p>
 * The following code fragment, in which <code>m</code> is a
 * <code>SOAPMessage</code> object and <code>contentStringl</code> is a
 * <code>String</code>, creates an instance of <code>AttachmentPart</code>,
 * sets the <code>AttachmentPart</code> object with some content and
 * header information, and adds the <code>AttachmentPart</code> object to
 * the <code>SOAPMessage</code> object.
 * <PRE>
```

```
 *     AttachmentPart ap1 = m.createAttachmentPart();
 *     ap1.setContent(contentString1, "text/plain");
 *     m.addAttachmentPart(ap1);
 * </PRE>
 *
 *
 * <p>
 * The following code fragment creates and adds a second
 * <code>AttachmentPart</code> instance to the same message. <code>jpegData</code>
 * is a binary byte buffer representing the jpeg file.
 * <PRE>
 *     AttachmentPart ap2 = m.createAttachmentPart();
 *     byte[] jpegData =  ...;
 *     ap2.setContent(new ByteArrayInputStream(jpegData), "image/jpeg");
 *     m.addAttachmentPart(ap2);
 * </PRE>
 * <p>
 * The <code>getContent</code> method retrieves the contents and header from
 * an <code>AttachmentPart</code> object. Depending on the
 * <code>DataContentHandler</code> objects present, the returned
 * <code>Object</code> can either be a typed Java object corresponding
 * to the MIME type or an <code>InputStream</code> object that contains the
 * content as bytes.
 * <PRE>
 *     String content1 = ap1.getContent();
 *     java.io.InputStream content2 = ap2.getContent();
 * </PRE>
 *
 * The method <code>clearContent</code> removes all the content from an
 * <code>AttachmentPart</code> object but does not affect its header information.
 * <PRE>
 *     ap1.clearContent();
 * </PRE>
 */

public abstract class AttachmentPart {
    /**
     * Returns the number of bytes in this <code>AttachmentPart</code>
     * object.
     *
     * @return the size of this <code>AttachmentPart</code> object in bytes
     *         or -1 if the size cannot be determined
     * @exception SOAPException if the content of this attachment is
     *            corrupted of if there was an exception while trying
     *            to determine the size.
     */
    public abstract int getSize() throws SOAPException;

    /**
     * Clears out the content of this <code>AttachmentPart</code> object.
     * The MIME header portion is left untouched.
     */
    public abstract void clearContent();
```

```
    /**
     * Gets the content of this <code>AttachmentPart</code> object as a Java
     * object. The type of the returned Java object depends on (1) the
     * <code>DataContentHandler</code> object that is used to interpret the bytes
     * and (2) the <code>Content-Type</code> given in the header.
     * <p>
     * For the MIME content types "text/plain", "text/html" and "text/xml", the
     * <code>DataContentHandler</code> object does the conversions to and
     * from the Java types corresponding to the MIME types.
     * For other MIME types,the <code>DataContentHandler</code> object
     * can return an <code>InputStream</code> object that contains the content data
     * as raw bytes.
     * <p>
     * A SAAJ-compliant implementation must, as a minimum, return a
     * <code>java.lang.String</code> object corresponding to any content
     * stream with a <code>Content-Type</code> value of
     * <code>text/plain</code>, a
     * <code>javax.xml.transform.stream.StreamSource</code> object corresponding to a
     * content stream with a <code>Content-Type</code> value of
     * <code>text/xml</code>, a <code>java.awt.Image</code> object
     * corresponding to a content stream with a
     * <code>Content-Type</code> value of <code>image/gif</code> or
     * <code>image/jpeg</code>.  For those content types that an
     * installed <code>DataContentHandler</code> object does not understand, the
     * <code>DataContentHandler</code> object is required to return a
     * <code>java.io.InputStream</code> object with the raw bytes.
     *
     * @return a Java object with the content of this <code>AttachmentPart</code>
     *         object
     *
     * @exception SOAPException if there is no content set into this
     *            <code>AttachmentPart</code> object or if there was a data
     *            transformation error
     */
    public abstract Object getContent() throws SOAPException;

    /**
     * Gets the content of this <code>AttachmentPart</code> object as an
     * InputStream as if a call had been made to <code>getContent</code> and no
     * <code>DataContentHandler</code> had been registered for the
     * <code>content-type</code> of this <code>AttachmentPart</code>.
     *<p>
     * Note that reading from the returned InputStream would result in consuming
     * the data in the stream. It is the responsibility of the caller to reset
     * the InputStream appropriately before calling a Subsequent API. If a copy
     * of the raw attachment content is required then the {@link #getRawContentBytes} API
     * should be used instead.
     *
     * @return an <code>InputStream</code> from which the raw data contained by
     *      the <code>AttachmentPart</code> can be accessed.
     *
     * @throws SOAPException if there is no content set into this
```

```
     *      <code>AttachmentPart</code> object or if there was a data
     *      transformation error.
     *
     * @since SAAJ 1.3
     * @see #getRawContentBytes
     */
    public abstract InputStream getRawContent() throws SOAPException;

    /**
     * Gets the content of this <code>AttachmentPart</code> object as a
     * byte[] array as if a call had been made to <code>getContent</code> and no
     * <code>DataContentHandler</code> had been registered for the
     * <code>content-type</code> of this <code>AttachmentPart</code>.
     *
     * @return a <code>byte[]</code> array containing the raw data of the
     *      <code>AttachmentPart</code>.
     *
     * @throws SOAPException if there is no content set into this
     *      <code>AttachmentPart</code> object or if there was a data
     *      transformation error.
     *
     * @since SAAJ 1.3
     */
    public abstract byte[] getRawContentBytes() throws SOAPException;

    /**
     * Returns an <code>InputStream</code> which can be used to obtain the
     * content of <code>AttachmentPart</code>  as Base64 encoded
     * character data, this method would base64 encode the raw bytes
     * of the attachment and return.
     *
     * @return an <code>InputStream</code> from which the Base64 encoded
     *       <code>AttachmentPart</code> can be read.
     *
     * @throws SOAPException if there is no content set into this
     *      <code>AttachmentPart</code> object or if there was a data
     *      transformation error.
     *
     * @since SAAJ 1.3
     */
    public abstract InputStream getBase64Content() throws SOAPException;

    /**
     * Sets the content of this attachment part to that of the given
     * <code>Object</code> and sets the value of the <code>Content-Type</code>
     * header to the given type. The type of the
     * <code>Object</code> should correspond to the value given for the
     * <code>Content-Type</code>. This depends on the particular
     * set of <code>DataContentHandler</code> objects in use.
     *
     *
     * @param object the Java object that makes up the content for
     *               this attachment part
```

```
     * @param contentType the MIME string that specifies the type of
     *                  the content
     *
     * @exception IllegalArgumentException may be thrown if the contentType
     *            does not match the type of the content object, or if there
     *            was no <code>DataContentHandler</code> object for this
     *            content object
     *
     * @see #getContent
     */
    public abstract void setContent(Object object, String contentType);

    /**
     * Sets the content of this attachment part to that contained by the
     * <code>InputStream</code> <code>content</code> and sets the value of the
     * <code>Content-Type</code> header to the value contained in
     * <code>contentType</code>.
     * <P>
     *  A subsequent call to getSize() may not be an exact measure
     *  of the content size.
     *
     * @param content the raw data to add to the attachment part
     * @param contentType the value to set into the <code>Content-Type</code>
     * header
     *
     * @exception SOAPException if an there is an error in setting the content
     * @exception NullPointerException if <code>content</code> is null
     * @since SAAJ 1.3
     */
    public abstract void setRawContent(InputStream content, String contentType) throws
        SOAPException;

    /**
     * Sets the content of this attachment part to that contained by the
     * <code>byte[]</code> array <code>content</code> and sets the value of the
     * <code>Content-Type</code> header to the value contained in
     * <code>contentType</code>.
     *
     * @param content the raw data to add to the attachment part
     * @param contentType the value to set into the <code>Content-Type</code>
     * header
     * @param offset the offset in the byte array of the content
     * @param len the number of bytes that form the content
     *
     * @exception SOAPException if an there is an error in setting the content
     * or content is null
     * @since SAAJ 1.3
     */
    public abstract void setRawContentBytes(
        byte[] content, int offset, int len,  String contentType)
        throws SOAPException;
```

```
    /**
     * Sets the content of this attachment part from the Base64 source
     * <code>InputStream</code>  and sets the value of the
     * <code>Content-Type</code> header to the value contained in
     * <code>contentType</code>, This method would first decode the base64
     * input and write the resulting raw bytes to the attachment.
     * <P>
     *  A subsequent call to getSize() may not be an exact measure
     *  of the content size.
     *
     * @param content the base64 encoded data to add to the attachment part
     * @param contentType the value to set into the <code>Content-Type</code>
     * header
     *
     * @exception SOAPException if an there is an error in setting the content
     * @exception NullPointerException if <code>content</code> is null
     *
     * @since SAAJ 1.3
     */
    public abstract void setBase64Content(
        InputStream content, String contentType) throws SOAPException;


    /**
     * Gets the <code>DataHandler</code> object for this <code>AttachmentPart</code>
     * object.
     *
     * @return the <code>DataHandler</code> object associated with this
     *         <code>AttachmentPart</code> object
     *
     * @exception SOAPException if there is no data in
     * this <code>AttachmentPart</code> object
     */
    public abstract DataHandler getDataHandler()
        throws SOAPException;

    /**
     * Sets the given <code>DataHandler</code> object as the data handler
     * for this <code>AttachmentPart</code> object. Typically, on an incoming
     * message, the data handler is automatically set. When
     * a message is being created and populated with content, the
     * <code>setDataHandler</code> method can be used to get data from
     * various data sources into the message.
     *
     * @param dataHandler the <code>DataHandler</code> object to be set
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified <code>DataHandler</code> object
     */
    public abstract void setDataHandler(DataHandler dataHandler);


    /**
```

```
     * Gets the value of the MIME header whose name is "Content-ID".
     *
     * @return a <code>String</code> giving the value of the
     *          "Content-ID" header or <code>null</code> if there
     *          is none
     * @see #setContentId
     */
    public String getContentId() {
        String[] values = getMimeHeader("Content-ID");
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    /**
     * Gets the value of the MIME header whose name is "Content-Location".
     *
     * @return a <code>String</code> giving the value of the
     *          "Content-Location" header or <code>null</code> if there
     *          is none
     */
    public String getContentLocation() {
        String[] values = getMimeHeader("Content-Location");
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    /**
     * Gets the value of the MIME header whose name is "Content-Type".
     *
     * @return a <code>String</code> giving the value of the
     *          "Content-Type" header or <code>null</code> if there
     *          is none
     */
    public String getContentType() {
        String[] values = getMimeHeader("Content-Type");
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    /**
     * Sets the MIME header whose name is "Content-ID" with the given value.
     *
     * @param contentId a <code>String</code> giving the value of the
     *          "Content-ID" header
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified <code>contentId</code> value
     * @see #getContentId
     */
    public void setContentId(String contentId)
```

```
    {
        setMimeHeader("Content-ID", contentId);
    }


    /**
     * Sets the MIME header whose name is "Content-Location" with the given value.
     *
     *
     * @param contentLocation a <code>String</code> giving the value of the
     *          "Content-Location" header
     * @exception IllegalArgumentException if there was a problem with
     *            the specified content location
     */
    public void setContentLocation(String contentLocation)
    {
        setMimeHeader("Content-Location", contentLocation);
    }

    /**
     * Sets the MIME header whose name is "Content-Type" with the given value.
     *
     * @param contentType a <code>String</code> giving the value of the
     *          "Content-Type" header
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified content type
     */
    public void setContentType(String contentType)
    {
        setMimeHeader("Content-Type", contentType);
    }

    /**
     * Removes all MIME headers that match the given name.
     *
     * @param header the string name of the MIME header/s to
     *               be removed
     */
    public abstract void removeMimeHeader(String header);

    /**
     * Removes all the MIME header entries.
     */
    public abstract void removeAllMimeHeaders();


    /**
     * Gets all the values of the header identified by the given
     * <code>String</code>.
     *
     * @param name the name of the header; example: "Content-Type"
     * @return a <code>String</code> array giving the value for the
```

```
     *         specified header
     * @see #setMimeHeader
     */
    public abstract String[] getMimeHeader(String name);


    /**
     * Changes the first header entry that matches the given name
     * to the given value, adding a new header if no existing header
     * matches. This method also removes all matching headers but the first. <p>
     *
     * Note that RFC822 headers can only contain US-ASCII characters.
     *
     * @param   name    a <code>String</code> giving the name of the header
     *                  for which to search
     * @param   value   a <code>String</code> giving the value to be set for
     *                  the header whose name matches the given name
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified mime header name or value
     */
    public abstract void setMimeHeader(String name, String value);


    /**
     * Adds a MIME header with the specified name and value to this
     * <code>AttachmentPart</code> object.
     * <p>
     * Note that RFC822 headers can contain only US-ASCII characters.
     *
     * @param   name    a <code>String</code> giving the name of the header
     *                  to be added
     * @param   value   a <code>String</code> giving the value of the header
     *                  to be added
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified mime header name or value
     */
    public abstract void addMimeHeader(String name, String value);

    /**
     * Retrieves all the headers for this <code>AttachmentPart</code> object
     * as an iterator over the <code>MimeHeader</code> objects.
     *
     * @return  an <code>Iterator</code> object with all of the Mime
     *          headers for this <code>AttachmentPart</code> object
     */
    public abstract Iterator getAllMimeHeaders();

    /**
     * Retrieves all <code>MimeHeader</code> objects that match a name in
     * the given array.
     *
```

```
     * @param names a <code>String</code> array with the name(s) of the
     *        MIME headers to be returned
     * @return  all of the MIME headers that match one of the names in the
     *           given array as an <code>Iterator</code> object
     */
    public abstract Iterator getMatchingMimeHeaders(String[] names);

    /**
     * Retrieves all <code>MimeHeader</code> objects whose name does
     * not match a name in the given array.
     *
     * @param names a <code>String</code> array with the name(s) of the
     *        MIME headers not to be returned
     * @return  all of the MIME headers in this <code>AttachmentPart</code> object
     *          except those that match one of the names in the
     *           given array.  The nonmatching MIME headers are returned as an
     *           <code>Iterator</code> object.
     */
    public abstract Iterator getNonMatchingMimeHeaders(String[] names);
}
```

17.2 Detail.java

Secuencia 163: javax/xml/soap/Detail.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.util.Iterator;

```

```
import javax.xml.namespace.QName;

/**
 * A container for <code>DetailEntry</code> objects. <code>DetailEntry</code>
 * objects give detailed error information that is application-specific and
 * related to the <code>SOAPBody</code> object that contains it.
 *<P>
 * A <code>Detail</code> object, which is part of a <code>SOAPFault</code>
 * object, can be retrieved using the method <code>SOAPFault.getDetail</code>.
 * The <code>Detail</code> interface provides two methods. One creates a new
 * <code>DetailEntry</code> object and also automatically adds it to
 * the <code>Detail</code> object. The second method gets a list of the
 * <code>DetailEntry</code> objects contained in a <code>Detail</code>
 * object.
 * <P>
 * The following code fragment, in which <i>sf</i> is a <code>SOAPFault</code>
 * object, gets its <code>Detail</code> object (<i>d</i>), adds a new
 * <code>DetailEntry</code> object to <i>d</i>, and then gets a list of all the
 * <code>DetailEntry</code> objects in <i>d</i>. The code also creates a
 * <code>Name</code> object to pass to the method <code>addDetailEntry</code>.
 * The variable <i>se</i>, used to create the <code>Name</code> object,
 * is a <code>SOAPEnvelope</code> object.
 * <PRE>
 *    Detail d = sf.getDetail();
 *    Name name = se.createName("GetLastTradePrice", "WOMBAT",
 *                              "http://www.wombat.org/trader");
 *    d.addDetailEntry(name);
 *    Iterator it = d.getDetailEntries();
 * </PRE>
 */
public interface Detail extends SOAPFaultElement {

    /**
     * Creates a new <code>DetailEntry</code> object with the given
     * name and adds it to this <code>Detail</code> object.
     *
     * @param name a <code>Name</code> object identifying the
     *         new <code>DetailEntry</code> object
     *
     * @exception SOAPException thrown when there is a problem in adding a
     * DetailEntry object to this Detail object.
     *
     * @see Detail#addDetailEntry(QName qname)
     */
    public DetailEntry addDetailEntry(Name name) throws SOAPException;

    /**
     * Creates a new <code>DetailEntry</code> object with the given
     * QName and adds it to this <code>Detail</code> object. This method
     * is the preferred over the one using Name.
     *
     * @param qname a <code>QName</code> object identifying the
     *         new <code>DetailEntry</code> object
```

```
     *
     * @exception SOAPException thrown when there is a problem in adding a
     * DetailEntry object to this Detail object.
     *
     * @see Detail#addDetailEntry(Name name)
     * @since SAAJ 1.3
     */
    public DetailEntry addDetailEntry(QName qname) throws SOAPException;

    /**
     * Gets an Iterator over all of the <code>DetailEntry</code>s in this <code>Detail</code>
         object.
     *
     * @return an <code>Iterator</code> object over the <code>DetailEntry</code>
     *             objects in this <code>Detail</code> object
     */
    public Iterator getDetailEntries();
}
```

17.3 DetailEntry.java

Secuencia 164: javax/xml/soap/DetailEntry.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * The content for a <code>Detail</code> object, giving details for
 * a <code>SOAPFault</code> object.  A <code>DetailEntry</code> object,
 * which carries information about errors related to the <code>SOAPBody</code>
```

```
 * object that contains it, is application-specific.
 */
public interface DetailEntry extends SOAPElement {

}
```

17.4 FactoryFinder.java

Secuencia 165: javax/xml/soap/FactoryFinder.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.io.*;
import java.util.Properties;


class FactoryFinder {

    /**
     * Creates an instance of the specified class using the specified
     * <code>ClassLoader</code> object.
     *
     * @exception SOAPException if the given class could not be found
     *            or could not be instantiated
     */
    private static Object newInstance(String className,
                                      ClassLoader classLoader)
            throws SOAPException
    {
```

```
        try {
            Class spiClass = safeLoadClass(className, classLoader);
            return spiClass.newInstance();

        } catch (ClassNotFoundException x) {
            throw new SOAPException("Provider " + className + " not found", x);
        } catch (Exception x) {
            throw new SOAPException("Provider " + className + " could not be instantiated: " + x, x
                );
        }
    }

    /**
     * Finds the implementation <code>Class</code> object for the given
     * factory name, or null if that fails.
     * <P>
     * This method is package private so that this code can be shared.
     *
     * @return the <code>Class</code> object of the specified message factory;
     *         or <code>null</code>
     *
     * @param factoryId             the name of the factory to find, which is
     *                              a system property
     * @exception SOAPException if there is a SOAP error
     */
    static Object find(String factoryId)
            throws SOAPException
    {
        return find(factoryId, null, false);
    }

    /**
     * Finds the implementation <code>Class</code> object for the given
     * factory name, or if that fails, finds the <code>Class</code> object
     * for the given fallback class name. The arguments supplied must be
     * used in order. If using the first argument is successful, the second
     * one will not be used.
     * <P>
     * This method is package private so that this code can be shared.
     *
     * @return the <code>Class</code> object of the specified message factory;
     *         may be <code>null</code>
     *
     * @param factoryId             the name of the factory to find, which is
     *                              a system property
     * @param fallbackClassName     the implementation class name, which is
     *                              to be used only if nothing else
     *                              is found; <code>null</code> to indicate that
     *                              there is no fallback class name
     * @exception SOAPException if there is a SOAP error
     */
    static Object find(String factoryId, String fallbackClassName)
            throws SOAPException
```

```
    {
        return find(factoryId, fallbackClassName, true);
    }

    /**
     * Finds the implementation <code>Class</code> object for the given
     * factory name, or if that fails, finds the <code>Class</code> object
     * for the given default class name, but only if <code>tryFallback</code>
     * is <code>true</code>.  The arguments supplied must be used in order
     * If using the first argument is successful, the second one will not
     * be used.  Note the default class name may be needed even if fallback
     * is not to be attempted, so certain error conditions can be handled.
     * <P>
     * This method is package private so that this code can be shared.
     *
     * @return the <code>Class</code> object of the specified message factory;
     *         may not be <code>null</code>
     *
     * @param factoryId             the name of the factory to find, which is
     *                              a system property
     * @param defaultClassName      the implementation class name, which is
     *                              to be used only if nothing else
     *                              is found; <code>null</code> to indicate
     *                              that there is no default class name
     * @param tryFallback           whether to try the default class as a
     *                              fallback
     * @exception SOAPException if there is a SOAP error
     */
    static Object find(String factoryId, String defaultClassName,
                       boolean tryFallback) throws SOAPException {
        ClassLoader classLoader;
        try {
            classLoader = Thread.currentThread().getContextClassLoader();
        } catch (Exception x) {
            throw new SOAPException(x.toString(), x);
        }

        // Use the system property first
        try {
            String systemProp =
                    System.getProperty( factoryId );
            if( systemProp!=null) {
                return newInstance(systemProp, classLoader);
            }
        } catch (SecurityException se) {
        }

        // try to read from $java.home/lib/jaxm.properties
        try {
            String javah=System.getProperty( "java.home" );
            String configFile = javah + File.separator +
                    "lib" + File.separator + "jaxm.properties";
            File f=new File( configFile );
```

```
            if( f.exists()) {
                Properties props=new Properties();
                props.load( new FileInputStream(f));
                String factoryClassName = props.getProperty(factoryId);
                return newInstance(factoryClassName, classLoader);
            }
        } catch(Exception ex ) {
        }

        String serviceId = "META-INF/services/" + factoryId;
        // try to find services in CLASSPATH
        try {
            InputStream is=null;
            if (classLoader == null) {
                is=ClassLoader.getSystemResourceAsStream(serviceId);
            } else {
                is=classLoader.getResourceAsStream(serviceId);
            }

            if( is!=null ) {
                BufferedReader rd =
                        new BufferedReader(new InputStreamReader(is, "UTF-8"));

                String factoryClassName = rd.readLine();
                rd.close();

                if (factoryClassName != null &&
                        ! "".equals(factoryClassName)) {
                    return newInstance(factoryClassName, classLoader);
                }
            }
        } catch( Exception ex ) {
        }

        // If not found and fallback should not be tried, return a null result.
        if (!tryFallback)
            return null;

        // We didn't find the class through the usual means so try the default
        // (built in) factory if specified.
        if (defaultClassName == null) {
            throw new SOAPException(
                    "Provider for " + factoryId + " cannot be found", null);
        }
        return newInstance(defaultClassName, classLoader);
    }

    /**
     * Loads the class, provided that the calling thread has an access to the
     * class being loaded. If this is the specified default factory class and it
     * is restricted by package.access we get a SecurityException and can do a
     * Class.forName() on it so it will be loaded by the bootstrap class loader.
     */
```

```
    private static Class safeLoadClass(String className,
                                       ClassLoader classLoader)
            throws ClassNotFoundException {
        try {
            // make sure that the current thread has an access to the package of the given name.
            SecurityManager s = System.getSecurityManager();
            if (s != null) {
                int i = className.lastIndexOf('.');
                if (i != -1) {
                    s.checkPackageAccess(className.substring(0, i));
                }
            }

            if (classLoader == null)
                return Class.forName(className);
            else
                return classLoader.loadClass(className);
        } catch (SecurityException se) {
            // (only) default implementation can be loaded
            // using bootstrap class loader:
            if (isDefaultImplementation(className))
                return Class.forName(className);

            throw se;
        }
    }

    private static boolean isDefaultImplementation(String className) {
        return MessageFactory.DEFAULT_MESSAGE_FACTORY.equals(className) ||
                SOAPFactory.DEFAULT_SOAP_FACTORY.equals(className) ||
                SOAPConnectionFactory.DEFAULT_SOAP_CONNECTION_FACTORY.equals(className) ||
                SAAJMetaFactory.DEFAULT_META_FACTORY_CLASS.equals(className);
    }
}
```

17.5 MessageFactory.java

Secuencia 166: javax/xml/soap/MessageFactory.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;


import java.io.IOException;
import java.io.InputStream;

/**
 * A factory for creating <code>SOAPMessage</code> objects.
 * <P>
 * A SAAJ client can create a <code>MessageFactory</code> object
 * using the method <code>newInstance</code>, as shown in the following
 * lines of code.
 * <PRE>
 *       MessageFactory mf = MessageFactory.newInstance();
 *       MessageFactory mf12 = MessageFactory.newInstance(SOAPConstants.SOAP_1_2_PROTOCOL);
 * </PRE>
 * <P>
 * All <code>MessageFactory</code> objects, regardless of how they are
 * created, will produce <code>SOAPMessage</code> objects that
 * have the following elements by default:
 * <UL>
 *  <LI>A <code>SOAPPart</code> object
 *  <LI>A <code>SOAPEnvelope</code> object
 *  <LI>A <code>SOAPBody</code> object
 *  <LI>A <code>SOAPHeader</code> object
 * </UL>
 * In some cases, specialized MessageFactory objects may be obtained that produce messages
 * prepopulated with additional entries in the <code>SOAPHeader</code> object and the
 * <code>SOAPBody</code> object.
 * The content of a new <code>SOAPMessage</code> object depends on which of the two
 * <code>MessageFactory</code> methods is used to create it.
 * <UL>
 *  <LI><code>createMessage()</code> <BR>
 *      This is the method clients would normally use to create a request message.
 *  <LI><code>createMessage(MimeHeaders, java.io.InputStream)</code> -- message has
 *       content from the <code>InputStream</code> object and headers from the
 *       <code>MimeHeaders</code> object <BR>
 *        This method can be used internally by a service implementation to
 *        create a message that is a response to a request.
 * </UL>
 */
public abstract class MessageFactory {

```

```java
    static final String DEFAULT_MESSAGE_FACTORY
        = "com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl";

    static private final String MESSAGE_FACTORY_PROPERTY
        = "javax.xml.soap.MessageFactory";

    /**
     * Creates a new <code>MessageFactory</code> object that is an instance
     * of the default implementation (SOAP 1.1),
     *
     * This method uses the following ordered lookup procedure to determine the MessageFactory implementation class to load:
     * <UL>
     *  <LI> Use the javax.xml.soap.MessageFactory system property.
     *  <LI> Use the properties file "lib/jaxm.properties" in the JRE directory. This configuration file is in standard
     * java.util.Properties format and contains the fully qualified name of the implementation class with the key being the
     * system property defined above.
     *  <LI> Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API
     * will look for a classname in the file META-INF/services/javax.xml.soap.MessageFactory in jars available to the runtime.
     *  <LI> Use the SAAJMetaFactory instance to locate the MessageFactory implementation class.
     * </UL>

     *
     * @return a new instance of a <code>MessageFactory</code>
     *
     * @exception SOAPException if there was an error in creating the
     *            default implementation of the
     *            <code>MessageFactory</code>.
     * @see SAAJMetaFactory
     */

    public static MessageFactory newInstance() throws SOAPException {


        try {
            MessageFactory factory = (MessageFactory) FactoryFinder.find(
                    MESSAGE_FACTORY_PROPERTY,
                    DEFAULT_MESSAGE_FACTORY,
                    false);

            if (factory != null) {
                return factory;
            }
            return newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

        } catch (Exception ex) {
            throw new SOAPException(
                    "Unable to create message factory for SOAP: "
                                +ex.getMessage());
```

```
        }

    }

    /**
     * Creates a new <code>MessageFactory</code> object that is an instance
     * of the specified implementation.  May be a dynamic message factory,
     * a SOAP 1.1 message factory, or a SOAP 1.2 message factory. A dynamic
     * message factory creates messages based on the MIME headers specified
     * as arguments to the <code>createMessage</code> method.
     *
     * This method uses the SAAJMetaFactory to locate the implementation class
     * and create the MessageFactory instance.
     *
     * @return a new instance of a <code>MessageFactory</code>
     *
     * @param protocol  a string constant representing the class of the
     *                  specified message factory implementation. May be
     *                  either <code>DYNAMIC_SOAP_PROTOCOL</code>,
     *                  <code>DEFAULT_SOAP_PROTOCOL</code> (which is the same
     *                  as) <code>SOAP_1_1_PROTOCOL</code>, or
     *                  <code>SOAP_1_2_PROTOCOL</code>.
     *
     * @exception SOAPException if there was an error in creating the
     *            specified implementation of  <code>MessageFactory</code>.
     * @see SAAJMetaFactory
     * @since SAAJ 1.3
     */
    public static MessageFactory newInstance(String protocol) throws SOAPException {
        return SAAJMetaFactory.getInstance().newMessageFactory(protocol);
    }

    /**
     * Creates a new <code>SOAPMessage</code> object with the default
     * <code>SOAPPart</code>, <code>SOAPEnvelope</code>, <code>SOAPBody</code>,
     * and <code>SOAPHeader</code> objects. Profile-specific message factories
     * can choose to prepopulate the <code>SOAPMessage</code> object with
     * profile-specific headers.
     * <P>
     * Content can be added to this message's <code>SOAPPart</code> object, and
     * the message can be sent "as is" when a message containing only a SOAP part
     * is sufficient. Otherwise, the <code>SOAPMessage</code> object needs
     * to create one or more <code>AttachmentPart</code> objects and
     * add them to itself. Any content that is not in XML format must be
     * in an <code>AttachmentPart</code> object.
     *
     * @return a new <code>SOAPMessage</code> object
     * @exception SOAPException if a SOAP error occurs
     * @exception UnsupportedOperationException if the protocol of this
     *      <code>MessageFactory</code> instance is <code>DYNAMIC_SOAP_PROTOCOL</code>
     */
    public abstract SOAPMessage createMessage()
        throws SOAPException;
```

```

    /**
     * Internalizes the contents of the given <code>InputStream</code> object into a
     * new <code>SOAPMessage</code> object and returns the <code>SOAPMessage</code>
     * object.
     *
     * @param in the <code>InputStream</code> object that contains the data
     *           for a message
     * @param headers the transport-specific headers passed to the
     *        message in a transport-independent fashion for creation of the
     *        message
     * @return a new <code>SOAPMessage</code> object containing the data from
     *         the given <code>InputStream</code> object
     *
     * @exception IOException if there is a problem in reading data from
     *            the input stream
     *
     * @exception SOAPException may be thrown if the message is invalid
     *
     * @exception IllegalArgumentException if the <code>MessageFactory</code>
     *      requires one or more MIME headers to be present in the
     *      <code>headers</code> parameter and they are missing.
     *      <code>MessageFactory</code> implementations for
     *      <code>SOAP_1_1_PROTOCOL</code> or
     *      <code>SOAP_1_2_PROTOCOL</code> must not throw
     *      <code>IllegalArgumentException</code> for this reason.
     */
    public abstract SOAPMessage createMessage(MimeHeaders headers,
                                              InputStream in)
        throws IOException, SOAPException;
}
```

17.6 MimeHeader.java

Secuencia 167: javax/xml/soap/MimeHeader.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

package javax.xml.soap;


/**
 * An object that stores a MIME header name and its value. One or more
 * <code>MimeHeader</code> objects may be contained in a <code>MimeHeaders</code>
 * object.
 *
 * @see MimeHeaders
 */
public class MimeHeader {

   private String name;
   private String value;

   /**
    * Constructs a <code>MimeHeader</code> object initialized with the given
    * name and value.
    *
    * @param name a <code>String</code> giving the name of the header
    * @param value a <code>String</code> giving the value of the header
    */
    public MimeHeader(String name, String value) {
        this.name = name;
        this.value = value;
    }

    /**
     * Returns the name of this <code>MimeHeader</code> object.
     *
     * @return the name of the header as a <code>String</code>
     */
    public String getName() {
        return name;
    }

    /**
     * Returns the value of this <code>MimeHeader</code> object.
     *
     * @return  the value of the header as a <code>String</code>
     */
    public String getValue() {
        return value;
    }
}
```

17.7 MimeHeaders.java

Secuencia 168: javax/xml/soap/MimeHeaders.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.util.Iterator;
import java.util.Vector;

/**
 * A container for <code>MimeHeader</code> objects, which represent
 * the MIME headers present in a MIME part of a message.
 *
 * <p>This class is used primarily when an application wants to
 * retrieve specific attachments based on certain MIME headers and
 * values. This class will most likely be used by implementations of
 * <code>AttachmentPart</code> and other MIME dependent parts of the SAAJ
 * API.
 * @see SOAPMessage#getAttachments
 * @see AttachmentPart
 */
public class MimeHeaders {
    private Vector headers;

   /**
    * Constructs a default <code>MimeHeaders</code> object initialized with
    * an empty <code>Vector</code> object.
    */
    public MimeHeaders() {
```

```
        headers = new Vector();
    }

    /**
     * Returns all of the values for the specified header as an array of
     * <code>String</code> objects.
     *
     * @param   name the name of the header for which values will be returned
     * @return a <code>String</code> array with all of the values for the
     *         specified header
     * @see #setHeader
     */
    public String[] getHeader(String name) {
        Vector values = new Vector();

        for(int i = 0; i < headers.size(); i++) {
            MimeHeader hdr = (MimeHeader) headers.elementAt(i);
            if (hdr.getName().equalsIgnoreCase(name)
                && hdr.getValue() != null)
                values.addElement(hdr.getValue());
        }

        if (values.size() == 0)
            return null;

        String r[] = new String[values.size()];
        values.copyInto(r);
        return r;
    }

    /**
     * Replaces the current value of the first header entry whose name matches
     * the given name with the given value, adding a new header if no existing header
     * name matches. This method also removes all matching headers after the first one.
     * <P>
     * Note that RFC822 headers can contain only US-ASCII characters.
     *
     * @param   name a <code>String</code> with the name of the header for
     *          which to search
     * @param   value a <code>String</code> with the value that will replace the
     *          current value of the specified header
     *
     * @exception IllegalArgumentException if there was a problem in the
     * mime header name or the value being set
     * @see #getHeader
     */
    public void setHeader(String name, String value)
    {
        boolean found = false;

        if ((name == null) || name.equals(""))
            throw new IllegalArgumentException("Illegal MimeHeader name");

```

```
        for(int i = 0; i < headers.size(); i++) {
            MimeHeader hdr = (MimeHeader) headers.elementAt(i);
            if (hdr.getName().equalsIgnoreCase(name)) {
                if (!found) {
                    headers.setElementAt(new MimeHeader(hdr.getName(),
                                                        value), i);
                    found = true;
                }
                else
                    headers.removeElementAt(i--);
            }
        }

        if (!found)
            addHeader(name, value);
    }

    /**
     * Adds a <code>MimeHeader</code> object with the specified name and value
     * to this <code>MimeHeaders</code> object's list of headers.
     * <P>
     * Note that RFC822 headers can contain only US-ASCII characters.
     *
     * @param   name a <code>String</code> with the name of the header to
     *          be added
     * @param   value a <code>String</code> with the value of the header to
     *          be added
     *
     * @exception IllegalArgumentException if there was a problem in the
     * mime header name or value being added
     */
    public void addHeader(String name, String value)
    {
        if ((name == null) || name.equals(""))
            throw new IllegalArgumentException("Illegal MimeHeader name");

        int pos = headers.size();

        for(int i = pos - 1 ; i >= 0; i--) {
            MimeHeader hdr = (MimeHeader) headers.elementAt(i);
            if (hdr.getName().equalsIgnoreCase(name)) {
                headers.insertElementAt(new MimeHeader(name, value),
                                        i+1);
                return;
            }
        }
        headers.addElement(new MimeHeader(name, value));
    }

    /**
     * Remove all <code>MimeHeader</code> objects whose name matches the
     * given name.
     *
```

```
     * @param   name a <code>String</code> with the name of the header for
     *          which to search
     */
    public void removeHeader(String name) {
        for(int i = 0; i < headers.size(); i++) {
            MimeHeader hdr = (MimeHeader) headers.elementAt(i);
            if (hdr.getName().equalsIgnoreCase(name))
                headers.removeElementAt(i--);
        }
    }

    /**
     * Removes all the header entries from this <code>MimeHeaders</code> object.
     */
    public void removeAllHeaders() {
        headers.removeAllElements();
    }


    /**
     * Returns all the <code>MimeHeader</code>s in this <code>MimeHeaders</code> object.
     *
     * @return  an <code>Iterator</code> object over this <code>MimeHeaders</code>
     *          object's list of <code>MimeHeader</code> objects
     */
    public Iterator getAllHeaders() {
        return headers.iterator();
    }

    class MatchingIterator implements Iterator {
        private boolean match;
        private Iterator iterator;
        private String[] names;
        private Object nextHeader;

        MatchingIterator(String[] names, boolean match) {
            this.match = match;
            this.names = names;
            this.iterator = headers.iterator();
        }

        private Object nextMatch() {
        next:
            while (iterator.hasNext()) {
                MimeHeader hdr = (MimeHeader) iterator.next();

                if (names == null)
                    return match ? null : hdr;

                for(int i = 0; i < names.length; i++)
                    if (hdr.getName().equalsIgnoreCase(names[i]))
                        if (match)
                            return hdr;
```

```
                    else
                        continue next;
            if (!match)
                return hdr;
        }
        return null;
    }


    public boolean hasNext() {
        if (nextHeader == null)
            nextHeader = nextMatch();
        return nextHeader != null;
    }

    public Object next() {
        // hasNext should've prefetched the header for us,
        // return it.
        if (nextHeader != null) {
            Object ret = nextHeader;
            nextHeader = null;
            return ret;
        }
        if (hasNext())
            return nextHeader;
        return null;
    }

    public void remove() {
        iterator.remove();
    }
}


/**
 * Returns all the <code>MimeHeader</code> objects whose name matches
 * a name in the given array of names.
 *
 * @param names an array of <code>String</code> objects with the names
 *         for which to search
 * @return  an <code>Iterator</code> object over the <code>MimeHeader</code>
 *          objects whose name matches one of the names in the given list
 */
public Iterator getMatchingHeaders(String[] names) {
    return new MatchingIterator(names, true);
}

/**
 * Returns all of the <code>MimeHeader</code> objects whose name does not
 * match a name in the given array of names.
 *
 * @param names an array of <code>String</code> objects with the names
 *         for which to search
```

```
     * @return  an <code>Iterator</code> object over the <code>MimeHeader</code>
     *          objects whose name does not match one of the names in the given list
     */
    public Iterator getNonMatchingHeaders(String[] names) {
        return new MatchingIterator(names, false);
    }
}
```

17.8 Name.java

Secuencia 169: javax/xml/soap/Name.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A representation of an XML name.  This interface provides methods for
 * getting the local and namespace-qualified names and also for getting the
 * prefix associated with the namespace for the name. It is also possible
 * to get the URI of the namespace.
 * <P>
 * The following is an example of a namespace declaration in an element.
 * <PRE>
 *   &lt;wombat:GetLastTradePrice xmlns:wombat="http://www.wombat.org/trader"&gt;
 * </PRE>
 * ("xmlns" stands for "XML namespace".)
 * The following
 * shows what the methods in the <code>Name</code> interface will return.
 * <UL>
 *  <LI><code>getQualifiedName</code> will return "prefix:LocalName" =
```

```
 *      "WOMBAT:GetLastTradePrice"
 *  <LI><code>getURI</code> will return "http://www.wombat.org/trader"
 *  <LI><code>getLocalName</code> will return "GetLastTracePrice"
 *  <LI><code>getPrefix</code> will return "WOMBAT"
 * </UL>
 * <P>
 * XML namespaces are used to disambiguate SOAP identifiers from
 * application-specific identifiers.
 * <P>
 * <code>Name</code> objects are created using the method
 * <code>SOAPEnvelope.createName</code>, which has two versions.
 * One method creates <code>Name</code> objects with
 * a local name, a namespace prefix, and a namespace URI.
 *  and the second creates <code>Name</code> objects with just a local name.
 * The following line of
 * code, in which <i>se</i> is a <code>SOAPEnvelope</code> object, creates a new
 * <code>Name</code> object with all three.
 * <PRE>
 *     Name name = se.createName("GetLastTradePrice", "WOMBAT",
 *                               "http://www.wombat.org/trader");
 * </PRE>
 * The following line of code gives an example of how a <code>Name</code> object
 * can be used. The variable <i>element</i> is a <code>SOAPElement</code> object.
 * This code creates a new <code>SOAPElement</code> object with the given name and
 * adds it to <i>element</i>.
 * <PRE>
 *     element.addChildElement(name);
 * </PRE>
 * <P>
 * The <code>Name</code> interface may be deprecated in a future release of SAAJ
 * in favor of <code>javax.xml.namespace.QName<code>
 * @see SOAPEnvelope#createName(String, String, String) SOAPEnvelope.createName
 * @see SOAPFactory#createName(String, String, String) SOAPFactory.createName
 */
public interface Name {
    /**
     * Gets the local name part of the XML name that this <code>Name</code>
     * object represents.
     *
     * @return a string giving the local name
     */
    String getLocalName();

    /**
     * Gets the namespace-qualified name of the XML name that this
     * <code>Name</code> object represents.
     *
     * @return the namespace-qualified name as a string
     */
    String getQualifiedName();

    /**
     * Returns the prefix that was specified when this <code>Name</code> object
```

```
     * was initialized. This prefix is associated with the namespace for the XML
     * name that this <code>Name</code> object represents.
     *
     * @return the prefix as a string
     */
    String getPrefix();

    /**
     * Returns the URI of the namespace for the XML
     * name that this <code>Name</code> object represents.
     *
     * @return the URI as a string
     */
    String getURI();
}
```

17.9 Node.java

Secuencia 170: javax/xml/soap/Node.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A representation of a node (element) in an XML document.
 * This interface extnends the standard DOM Node interface with methods for
 * getting and setting the value of a node, for
 * getting and setting the parent of a node, and for removing a node.
 */
public interface Node extends org.w3c.dom.Node {
```

```
    /**
     * Returns the value of this node if this is a <code>Text</code> node or the
     * value of the immediate child of this node otherwise.
     * If there is an immediate child of this <code>Node</code> that it is a
     * <code>Text</code> node then it's value will be returned. If there is
     * more than one <code>Text</code> node then the value of the first
     * <code>Text</code> Node will be returned.
     * Otherwise <code>null</code> is returned.
     *
     * @return a <code>String</code> with the text of this node if this is a
     *          <code>Text</code> node or the text contained by the first
     *          immediate child of this <code>Node</code> object that is a
     *          <code>Text</code> object if such a child exists;
     *          <code>null</code> otherwise.
     */
    public String getValue();

    /**
     * If this is a Text node then this method will set its value,
     * otherwise it sets the value of  the immediate (Text) child of this node.
     * The value of the immediate child of this node can be set only if, there is
     * one child node and that node is a <code>Text</code> node, or if
     * there are no children in which case a child <code>Text</code> node will be
     * created.
     *
     * @exception IllegalStateException if the node is not a <code>Text</code>
     *              node and either has more than one child node or has a child
     *              node that is not a <code>Text</code> node.
     *
     * @since SAAJ 1.2
     */
    public void setValue(String value);

    /**
     * Sets the parent of this <code>Node</code> object to the given
     * <code>SOAPElement</code> object.
     *
     * @param parent the <code>SOAPElement</code> object to be set as
     *       the parent of this <code>Node</code> object
     *
     * @exception SOAPException if there is a problem in setting the
     *                          parent to the given element
     * @see #getParentElement
     */
    public void setParentElement(SOAPElement parent) throws SOAPException;

    /**
     * Returns the parent element of this <code>Node</code> object.
     * This method can throw an <code>UnsupportedOperationException</code>
     * if the tree is not kept in memory.
     *
     * @return the <code>SOAPElement</code> object that is the parent of
     *         this <code>Node</code> object or <code>null</code> if this
```

```
     *         <code>Node</code> object is root
     *
     * @exception UnsupportedOperationException if the whole tree is not
     *            kept in memory
     * @see #setParentElement
     */
    public SOAPElement getParentElement();

    /**
     * Removes this <code>Node</code> object from the tree.
     */
    public void detachNode();

    /**
     * Notifies the implementation that this <code>Node</code>
     * object is no longer being used by the application and that the
     * implementation is free to reuse this object for nodes that may
     * be created later.
     * <P>
     * Calling the method <code>recycleNode</code> implies that the method
     * <code>detachNode</code> has been called previously.
     */
    public void recycleNode();

}
```

17.10 SAAJMetaFactory.java

Secuencia 171: javax/xml/soap/SAAJMetaFactory.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```

package javax.xml.soap;

/**
* The access point for the implementation classes of the factories defined in the
* SAAJ API. All of the <code>newInstance</code> methods defined on factories in
* SAAJ 1.3 defer to instances of this class to do the actual object creation.
* The implementations of <code>newInstance()</code> methods (in SOAPFactory and MessageFactory)
* that existed in SAAJ 1.2 have been updated to also delegate to the SAAJMetaFactory when the SAAJ 1.2
* defined lookup fails to locate the Factory implementation class name.
*
* <p>
* SAAJMetaFactory is a service provider interface. There are no public methods on this
* class.
*
* @author SAAJ RI Development Team
* @since SAAJ 1.3
*/

public abstract class SAAJMetaFactory {
    static private final String META_FACTORY_CLASS_PROPERTY =
        "javax.xml.soap.MetaFactory";
    static final String DEFAULT_META_FACTORY_CLASS =
        "com.sun.xml.internal.messaging.saaj.soap.SAAJMetaFactoryImpl";

    /**
     * Creates a new instance of a concrete <code>SAAJMetaFactory</code> object.
     * The SAAJMetaFactory is an SPI, it pulls the creation of the other factories together into a
     * single place. Changing out the SAAJMetaFactory has the effect of changing out the entire SAAJ
     * implementation. Service providers provide the name of their <code>SAAJMetaFactory</code>
     * implementation.
     *
     * This method uses the following ordered lookup procedure to determine the SAAJMetaFactory implementation class to load:
     * <UL>
     *  <LI> Use the javax.xml.soap.MetaFactory system property.
     *  <LI> Use the properties file "lib/jaxm.properties" in the JRE directory. This configuration file is in standard
     * java.util.Properties format and contains the fully qualified name of the implementation class with the key being the
     * system property defined above.
     *  <LI> Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API
     * will look for a classname in the file META-INF/services/javax.xml.soap.MetaFactory in jars available to the runtime.
     *  <LI> Default to com.sun.xml.internal.messaging.saaj.soap.SAAJMetaFactoryImpl.
     * </UL>
     *
     * @return a concrete <code>SAAJMetaFactory</code> object
     * @exception SOAPException if there is an error in creating the <code>SAAJMetaFactory</code>
     */
```

```
    static SAAJMetaFactory getInstance() throws SOAPException {
            try {
                SAAJMetaFactory instance =
                    (SAAJMetaFactory) FactoryFinder.find(
                        META_FACTORY_CLASS_PROPERTY,
                        DEFAULT_META_FACTORY_CLASS);
                return instance;
            } catch (Exception e) {
                throw new SOAPException(
                    "Unable to create SAAJ meta-factory" + e.getMessage());
            }
    }

    protected SAAJMetaFactory() { }

     /**
      * Creates a <code>MessageFactory</code> object for
      * the given <code>String</code> protocol.
      *
      * @param protocol a <code>String</code> indicating the protocol
      * @exception SOAPException if there is an error in creating the
      *            MessageFactory
      * @see SOAPConstants#SOAP_1_1_PROTOCOL
      * @see SOAPConstants#SOAP_1_2_PROTOCOL
      * @see SOAPConstants#DYNAMIC_SOAP_PROTOCOL
      */
    protected abstract MessageFactory newMessageFactory(String protocol)
        throws SOAPException;

     /**
      * Creates a <code>SOAPFactory</code> object for
      * the given <code>String</code> protocol.
      *
      * @param protocol a <code>String</code> indicating the protocol
      * @exception SOAPException if there is an error in creating the
      *            SOAPFactory
      * @see SOAPConstants#SOAP_1_1_PROTOCOL
      * @see SOAPConstants#SOAP_1_2_PROTOCOL
      * @see SOAPConstants#DYNAMIC_SOAP_PROTOCOL
      */
    protected abstract SOAPFactory newSOAPFactory(String protocol)
        throws SOAPException;
}
```

17.11 SAAJResult.java

Secuencia 172: javax/xml/soap/SAAJResult.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import javax.xml.transform.dom.DOMResult;

/**
 * Acts as a holder for the results of a JAXP transformation or a JAXB
 * marshalling, in the form of a SAAJ tree. These results should be accessed
 * by using the {@link #getResult()} method. The {@link DOMResult#getNode()}
 * method should be avoided in almost all cases.
 *
 * @author XWS-Security Development Team
 *
 * @since SAAJ 1.3
 */
public class SAAJResult extends DOMResult {

    /**
     * Creates a <code>SAAJResult</code> that will present results in the form
     * of a SAAJ tree that supports the default (SOAP 1.1) protocol.
     * <p>
     * This kind of <code>SAAJResult</code> is meant for use in situations where the
     * results will be used as a parameter to a method that takes a parameter
     * whose type, such as <code>SOAPElement</code>, is drawn from the SAAJ
     * API. When used in a transformation, the results are populated into the
     * <code>SOAPPart</code> of a <code>SOAPMessage</code> that is created internally.
     * The <code>SOAPPart</code> returned by {@link DOMResult#getNode()}
     * is not guaranteed to be well-formed.
     *
     * @throws SOAPException if there is a problem creating a <code>SOAPMessage</code>
     *
     * @since SAAJ 1.3
     */
    public SAAJResult() throws SOAPException {
        this(MessageFactory.newInstance().createMessage());
```

```
    }

    /**
     * Creates a <code>SAAJResult</code> that will present results in the form
     * of a SAAJ tree that supports the specified protocol. The
     * <code>DYNAMIC_SOAP_PROTOCOL</code> is ambiguous in this context and will
     * cause this constructor to throw an <code>UnsupportedOperationException</code>.
     * <p>
     * This kind of <code>SAAJResult</code> is meant for use in situations where the
     * results will be used as a parameter to a method that takes a parameter
     * whose type, such as <code>SOAPElement</code>, is drawn from the SAAJ
     * API. When used in a transformation the results are populated into the
     * <code>SOAPPart</code> of a <code>SOAPMessage</code> that is created
     * internally. The <code>SOAPPart</code> returned by {@link DOMResult#getNode()}
     * is not guaranteed to be well-formed.
     *
     * @param protocol - the name of the SOAP protocol that the resulting SAAJ
     *                      tree should support
     *
     * @throws SOAPException if a <code>SOAPMessage</code> supporting the
     *             specified protocol cannot be created
     *
     * @since SAAJ 1.3
     */
    public SAAJResult(String protocol) throws SOAPException {
        this(MessageFactory.newInstance(protocol).createMessage());
    }

    /**
     * Creates a <code>SAAJResult</code> that will write the results into the
     * <code>SOAPPart</code> of the supplied <code>SOAPMessage</code>.
     * In the normal case these results will be written using DOM APIs and,
     * as a result, the finished <code>SOAPPart</code> will not be guaranteed
     * to be well-formed unless the data used to create it is also well formed.
     * When used in a transformation the validity of the <code>SOAPMessage</code>
     * after the transformation can be guaranteed only by means outside SAAJ
     * specification.
     *
     * @param message - the message whose <code>SOAPPart</code> will be
     *                  populated as a result of some transformation or
     *                  marshalling operation
     *
     * @since SAAJ 1.3
     */
    public SAAJResult(SOAPMessage message) {
        super(message.getSOAPPart());
    }

    /**
     * Creates a <code>SAAJResult</code> that will write the results as a
     * child node of the <code>SOAPElement</code> specified. In the normal
     * case these results will be written using DOM APIs and as a result may
     * invalidate the structure of the SAAJ tree. This kind of
```

```
     * <code>SAAJResult</code> should only be used when the validity of the
     * incoming data can be guaranteed by means outside of the SAAJ
     * specification.
     *
     * @param rootNode - the root to which the results will be appended
     *
     * @since SAAJ 1.3
     */
    public SAAJResult(SOAPElement rootNode) {
        super(rootNode);
    }


    /**
     * @return the resulting Tree that was created under the specified root Node.
     * @since SAAJ 1.3
     */
    public javax.xml.soap.Node getResult() {
        return (javax.xml.soap.Node)super.getNode().getFirstChild();
     }
}
```

17.12 SOAPBody.java

Secuencia 173: javax/xml/soap/SOAPBody.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.util.Locale;
```

```java

import org.w3c.dom.Document;

import javax.xml.namespace.QName;

/**
 * An object that represents the contents of the SOAP body
 * element in a SOAP message. A SOAP body element consists of XML data
 * that affects the way the application-specific content is processed.
 * <P>
 * A <code>SOAPBody</code> object contains <code>SOAPBodyElement</code>
 * objects, which have the content for the SOAP body.
 * A <code>SOAPFault</code> object, which carries status and/or
 * error information, is an example of a <code>SOAPBodyElement</code> object.
 *
 * @see SOAPFault
 */
public interface SOAPBody extends SOAPElement {

    /**
     * Creates a new <code>SOAPFault</code> object and adds it to
     * this <code>SOAPBody</code> object. The new <code>SOAPFault</code> will
     * have default values set for the mandatory child elements. The type of
     * the <code>SOAPFault</code> will be a SOAP 1.1 or a SOAP 1.2 <code>SOAPFault</code>
     * depending on the <code>protocol</code> specified while creating the
     * <code>MessageFactory</code> instance.
     * <p>
     * A <code>SOAPBody</code> may contain at most one <code>SOAPFault</code>
     * child element.
     *
     * @return the new <code>SOAPFault</code> object
     * @exception SOAPException if there is a SOAP error
     */
    public SOAPFault addFault() throws SOAPException;


    /**
     * Creates a new <code>SOAPFault</code> object and adds it to
     * this <code>SOAPBody</code> object. The type of the
     * <code>SOAPFault</code> will be a SOAP 1.1  or a SOAP 1.2
     * <code>SOAPFault</code> depending on the <code>protocol</code>
     * specified while creating the <code>MessageFactory</code> instance.
     * <p>
     * For SOAP 1.2 the <code>faultCode</code> parameter is the value of the
     * <i>Fault/Code/Value</i> element  and the <code>faultString</code> parameter
     * is the value of the <i>Fault/Reason/Text</i> element. For SOAP 1.1
     * the <code>faultCode</code> parameter is the value of the <code>faultcode</code>
     * element and the <code>faultString</code> parameter is the value of the <code>faultstring</code>
     * element.
     * <p>
     * A <code>SOAPBody</code> may contain at most one <code>SOAPFault</code>
     * child element.
```

```
     *
     * @param faultCode a <code>Name</code> object giving the fault
     *         code to be set; must be one of the fault codes defined in the Version
     *         of SOAP specification in use
     * @param faultString a <code>String</code> giving an explanation of
     *         the fault
     * @param locale a {@link java.util.Locale} object indicating
     *         the native language of the <code>faultString</code>
     * @return the new <code>SOAPFault</code> object
     * @exception SOAPException if there is a SOAP error
     * @see SOAPFault#setFaultCode
     * @see SOAPFault#setFaultString
     * @since SAAJ 1.2
     */
    public SOAPFault addFault(Name faultCode, String faultString, Locale locale) throws
        SOAPException;

    /**
     * Creates a new <code>SOAPFault</code> object and adds it to this
     * <code>SOAPBody</code> object. The type of the <code>SOAPFault</code>
     * will be a SOAP 1.1 or a SOAP 1.2 <code>SOAPFault</code> depending on
     * the <code>protocol</code> specified while creating the <code>MessageFactory</code>
     * instance.
     * <p>
     * For SOAP 1.2 the <code>faultCode</code> parameter is the value of the
     * <i>Fault/Code/Value</i> element  and the <code>faultString</code> parameter
     * is the value of the <i>Fault/Reason/Text</i> element. For SOAP 1.1
     * the <code>faultCode</code> parameter is the value of the <code>faultcode</code>
     * element and the <code>faultString</code> parameter is the value of the <code>faultstring</
        code>
     * element.
     * <p>
     * A <code>SOAPBody</code> may contain at most one <code>SOAPFault</code>
     * child element.
     *
     * @param faultCode
     *            a <code>QName</code> object giving the fault code to be
     *            set; must be one of the fault codes defined in the version
     *            of SOAP specification in use.
     * @param faultString
     *            a <code>String</code> giving an explanation of the fault
     * @param locale
     *            a {@link java.util.Locale Locale} object indicating the
     *            native language of the <code>faultString</code>
     * @return the new <code>SOAPFault</code> object
     * @exception SOAPException
     *                if there is a SOAP error
     * @see SOAPFault#setFaultCode
     * @see SOAPFault#setFaultString
     * @see SOAPBody#addFault(Name faultCode, String faultString, Locale locale)
     *
     * @since SAAJ 1.3
     */
```

```
    public SOAPFault addFault(QName faultCode, String faultString, Locale locale)
        throws SOAPException;

    /**
     * Creates a new  <code>SOAPFault</code> object and adds it to this
     * <code>SOAPBody</code> object. The type of the <code>SOAPFault</code>
     * will be a SOAP 1.1 or a SOAP 1.2 <code>SOAPFault</code> depending on
     * the <code>protocol</code> specified while creating the <code>MessageFactory</code>
     * instance.
     * <p>
     * For SOAP 1.2 the <code>faultCode</code> parameter is the value of the
     * <i>Fault/Code/Value</i> element  and the <code>faultString</code> parameter
     * is the value of the <i>Fault/Reason/Text</i> element. For SOAP 1.1
     * the <code>faultCode</code> parameter is the value of the <i>faultcode</i>
     * element and the <code>faultString</code> parameter is the value of the <i>faultstring</i>
     * element.
     * <p>
     * In case of a SOAP 1.2 fault, the default value for the mandatory <code>xml:lang</code>
     * attribute on the <i>Fault/Reason/Text</i> element will be set to
     * <code>java.util.Locale.getDefault()</code>
     * <p>
     * A <code>SOAPBody</code> may contain at most one <code>SOAPFault</code>
     * child element.
     *
     * @param faultCode
     *            a <code>Name</code> object giving the fault code to be set;
     *            must be one of the fault codes defined in the version of SOAP
     *            specification in use
     * @param faultString
     *            a <code>String</code> giving an explanation of the fault
     * @return the new <code>SOAPFault</code> object
     * @exception SOAPException
     *                if there is a SOAP error
     * @see SOAPFault#setFaultCode
     * @see SOAPFault#setFaultString
     * @since SAAJ 1.2
     */
    public SOAPFault addFault(Name faultCode, String faultString)
        throws SOAPException;

    /**
     * Creates a new <code>SOAPFault</code> object and adds it to this <code>SOAPBody</code>
     * object. The type of the <code>SOAPFault</code>
     * will be a SOAP 1.1 or a SOAP 1.2 <code>SOAPFault</code> depending on
     * the <code>protocol</code> specified while creating the <code>MessageFactory</code>
     * instance.
     * <p>
     * For SOAP 1.2 the <code>faultCode</code> parameter is the value of the
     * <i>Fault/Code/Value</i> element  and the <code>faultString</code> parameter
     * is the value of the <i>Fault/Reason/Text</i> element. For SOAP 1.1
     * the <code>faultCode</code> parameter is the value of the <i>faultcode</i>
     * element and the <code>faultString</code> parameter is the value of the <i>faultstring</i>
     * element.
```

```
     * <p>
     * In case of a SOAP 1.2 fault, the default value for the mandatory <code>xml:lang</code>
     * attribute on the <i>Fault/Reason/Text</i> element will be set to
     * <code>java.util.Locale.getDefault()</code>
     * <p>
     * A <code>SOAPBody</code> may contain at most one <code>SOAPFault</code>
     * child element
     *
     * @param faultCode
     *            a <code>QName</code> object giving the fault code to be
     *            set; must be one of the fault codes defined in the version
     *            of  SOAP specification in use
     * @param faultString
     *            a <code>String</code> giving an explanation of the fault
     * @return the new <code>SOAPFault</code> object
     * @exception SOAPException
     *                if there is a SOAP error
     * @see SOAPFault#setFaultCode
     * @see SOAPFault#setFaultString
     * @see SOAPBody#addFault(Name faultCode, String faultString)
     * @since SAAJ 1.3
     */
    public SOAPFault addFault(QName faultCode, String faultString)
        throws SOAPException;

    /**
     * Indicates whether a <code>SOAPFault</code> object exists in this
     * <code>SOAPBody</code> object.
     *
     * @return <code>true</code> if a <code>SOAPFault</code> object exists
     *         in this <code>SOAPBody</code> object; <code>false</code>
     *         otherwise
     */
    public boolean hasFault();

    /**
     * Returns the <code>SOAPFault</code> object in this <code>SOAPBody</code>
     * object.
     *
     * @return the <code>SOAPFault</code> object in this <code>SOAPBody</code>
     *         object if present, null otherwise.
     */
    public SOAPFault getFault();

    /**
     * Creates a new <code>SOAPBodyElement</code> object with the specified
     * name and adds it to this <code>SOAPBody</code> object.
     *
     * @param name
     *            a <code>Name</code> object with the name for the new <code>SOAPBodyElement</code>
     *            object
     * @return the new <code>SOAPBodyElement</code> object
     * @exception SOAPException
```

```
     *                if a SOAP error occurs
     * @see SOAPBody#addBodyElement(javax.xml.namespace.QName)
     */
    public SOAPBodyElement addBodyElement(Name name) throws SOAPException;


    /**
     * Creates a new <code>SOAPBodyElement</code> object with the specified
     * QName and adds it to this <code>SOAPBody</code> object.
     *
     * @param qname
     *            a <code>QName</code> object with the qname for the new
     *            <code>SOAPBodyElement</code> object
     * @return the new <code>SOAPBodyElement</code> object
     * @exception SOAPException
     *                if a SOAP error occurs
     * @see SOAPBody#addBodyElement(Name)
     * @since SAAJ 1.3
     */
    public SOAPBodyElement addBodyElement(QName qname) throws SOAPException;

    /**
     * Adds the root node of the DOM <code>{@link org.w3c.dom.Document}</code>
     * to this <code>SOAPBody</code> object.
     * <p>
     * Calling this method invalidates the <code>document</code> parameter.
     * The client application should discard all references to this <code>Document</code>
     * and its contents upon calling <code>addDocument</code>. The behavior
     * of an application that continues to use such references is undefined.
     *
     * @param document
     *            the <code>Document</code> object whose root node will be
     *            added to this <code>SOAPBody</code>.
     * @return the <code>SOAPBodyElement</code> that represents the root node
     *         that was added.
     * @exception SOAPException
     *                if the <code>Document</code> cannot be added
     * @since SAAJ 1.2
     */
    public SOAPBodyElement addDocument(org.w3c.dom.Document document)
        throws SOAPException;

    /**
     * Creates a new DOM <code>{@link org.w3c.dom.Document}</code> and sets
     * the first child of this <code>SOAPBody</code> as it's document
     * element. The child <code>SOAPElement</code> is removed as part of the
     * process.
     *
     * @return the <code>{@link org.w3c.dom.Document}</code> representation
     *         of the <code>SOAPBody</code> content.
     *
     * @exception SOAPException
     *                if there is not exactly one child <code>SOAPElement</code> of the <code>
```

```
     *              <code>SOAPBody</code>.
     *
     * @since SAAJ 1.3
     */
    public org.w3c.dom.Document extractContentAsDocument()
        throws SOAPException;
}
```

17.13 SOAPBodyElement.java

Secuencia 174: javax/xml/soap/SOAPBodyElement.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A <code>SOAPBodyElement</code> object represents the contents in
 * a <code>SOAPBody</code> object.  The <code>SOAPFault</code> interface
 * is a <code>SOAPBodyElement</code> object that has been defined.
 * <P>
 * A new <code>SOAPBodyElement</code> object can be created and added
 * to a <code>SOAPBody</code> object with the <code>SOAPBody</code>
 * method <code>addBodyElement</code>. In the following line of code,
 * <code>sb</code> is a <code>SOAPBody</code> object, and
 * <code>myName</code> is a <code>Name</code> object.
 * <PRE>
 *    SOAPBodyElement sbe = sb.addBodyElement(myName);
 * </PRE>
 */
public interface SOAPBodyElement extends SOAPElement {
```

```
}
```

17.14 SOAPConnection.java

Secuencia 175: javax/xml/soap/SOAPConnection.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;


/**
 * A point-to-point connection that a client can use for sending messages
 * directly to a remote party (represented by a URL, for instance).
 * <p>
 * The SOAPConnection class is optional. Some implementations may
 * not implement this interface in which case the call to
 * <code>SOAPConnectionFactory.newInstance()</code> (see below) will
 * throw an <code>UnsupportedOperationException</code>.
 * <p>
 * A client can obtain a <code>SOAPConnection</code> object using a
 * {@link SOAPConnectionFactory} object as in the following example:
 * <PRE>
 *      SOAPConnectionFactory factory = SOAPConnectionFactory.newInstance();
 *      SOAPConnection con = factory.createConnection();
 * </PRE>
 * A <code>SOAPConnection</code> object can be used to send messages
 * directly to a URL following the request/response paradigm.  That is,
 * messages are sent using the method <code>call</code>, which sends the
 * message and then waits until it gets a reply.
 */
```

```java
public abstract class SOAPConnection {

    /**
     * Sends the given message to the specified endpoint and blocks until
     * it has returned the response.
     *
     * @param request the <code>SOAPMessage</code> object to be sent
     * @param to an <code>Object</code> that identifies
     *         where the message should be sent. It is required to
     *         support Objects of type
     *         <code>java.lang.String</code>,
     *         <code>java.net.URL</code>, and when JAXM is present
     *         <code>javax.xml.messaging.URLEndpoint</code>
     *
     * @return the <code>SOAPMessage</code> object that is the response to the
     *         message that was sent
     * @throws SOAPException if there is a SOAP error
     */
    public abstract SOAPMessage call(SOAPMessage request,
                                     Object to) throws SOAPException;

    /**
     * Gets a message from a specific endpoint and blocks until it receives,
     *
     * @param to an <code>Object</code> that identifies where
     *                  the request should be sent. Objects of type
     *                 <code>java.lang.String</code> and
     *                 <code>java.net.URL</code> must be supported.
     *
     * @return the <code>SOAPMessage</code> object that is the response to the
     *                  get message request
     * @throws SOAPException if there is a SOAP error
     * @since SAAJ 1.3
     */
    public SOAPMessage get(Object to)
                               throws SOAPException {
        throw new UnsupportedOperationException("All subclasses of SOAPConnection must override get
            ()");
    }

    /**
     * Closes this <code>SOAPConnection</code> object.
     *
     * @throws SOAPException if there is a SOAP error
     */
    public abstract void close()
        throws SOAPException;
}
```

17.15 SOAPConnectionFactory.java

Secuencia 176: javax/xml/soap/SOAPConnectionFactory.java

```java
/*
```

```
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A factory for creating <code>SOAPConnection</code> objects. Implementation of this class
 * is optional. If <code>SOAPConnectionFactory.newInstance()</code> throws an
 * UnsupportedOperationException then the implementation does not support the
 * SAAJ communication infrastructure. Otherwise {@link SOAPConnection} objects
 * can be created by calling <code>createConnection()</code> on the newly
 * created <code>SOAPConnectionFactory</code> object.
 */
public abstract class SOAPConnectionFactory {
    /**
     * A constant representing the default value for a <code>SOAPConnection</code>
     * object. The default is the point-to-point SOAP connection.
     */
    static final String DEFAULT_SOAP_CONNECTION_FACTORY
        = "com.sun.xml.internal.messaging.saaj.client.p2p.HttpSOAPConnectionFactory";

    /**
     * A constant representing the <code>SOAPConnection</code> class.
     */
    static private final String SF_PROPERTY
        = "javax.xml.soap.SOAPConnectionFactory";

    /**
     * Creates an instance of the default
     * <code>SOAPConnectionFactory</code> object.
     *
     * @return a new instance of a default
```

```
     *         <code>SOAPConnectionFactory</code> object
     *
     * @exception SOAPException if there was an error creating the
     *            <code>SOAPConnectionFactory</code>
     *
     * @exception UnsupportedOperationException if newInstance is not
     * supported.
     */
    public static SOAPConnectionFactory newInstance()
        throws SOAPException, UnsupportedOperationException
    {
        try {
        return (SOAPConnectionFactory)
                FactoryFinder.find(SF_PROPERTY,
                                   DEFAULT_SOAP_CONNECTION_FACTORY);
        } catch (Exception ex) {
            throw new SOAPException("Unable to create SOAP connection factory: "
                                    +ex.getMessage());
        }
    }

    /**
     * Create a new <code>SOAPConnection</code>.
     *
     * @return the new <code>SOAPConnection</code> object.
     *
     * @exception SOAPException if there was an exception creating the
     * <code>SOAPConnection</code> object.
     */
    public abstract SOAPConnection createConnection()
        throws SOAPException;
}
```

17.16 SOAPConstants.java

Secuencia 177: javax/xml/soap/SOAPConstants.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import javax.xml.namespace.QName;

/**
 * The definition of constants pertaining to the SOAP protocol.
 */
public interface SOAPConstants {
    /**
     * Used to create <code>MessageFactory</code> instances that create
     * <code>SOAPMessages</code> whose concrete type is based on the
     * <code>Content-Type</code> MIME header passed to the
     * <code>createMessage</code> method. If no <code>Content-Type</code>
     * header is passed then the <code>createMessage</code> may throw an
     * <code>IllegalArgumentException</code> or, in the case of the no
     * argument version of <code>createMessage</code>, an
     * <code>UnsupportedOperationException</code>.
     *
     * @since  SAAJ 1.3
     */
    public static final String DYNAMIC_SOAP_PROTOCOL = "Dynamic Protocol";

    /**
     * Used to create <code>MessageFactory</code> instances that create
     * <code>SOAPMessages</code> whose behavior supports the SOAP 1.1  specification.
     *
     * @since  SAAJ 1.3
     */
    public static final String SOAP_1_1_PROTOCOL = "SOAP 1.1 Protocol";

    /**
     * Used to create <code>MessageFactory</code> instances that create
     * <code>SOAPMessages</code> whose behavior supports the SOAP 1.2
     * specification
     *
     * @since  SAAJ 1.3
     */
    public static final String SOAP_1_2_PROTOCOL = "SOAP 1.2 Protocol";

    /**
     * The default protocol: SOAP 1.1 for backwards compatibility.
     *
     * @since SAAJ 1.3
     */
    public static final String DEFAULT_SOAP_PROTOCOL = SOAP_1_1_PROTOCOL;
```

```

    /**
     * The namespace identifier for the SOAP 1.1 envelope.
     * @since SAAJ 1.3
     */
    public static final String
                URI_NS_SOAP_1_1_ENVELOPE = "http://schemas.xmlsoap.org/soap/envelope/";
    /**
     * The namespace identifier for the SOAP 1.2 envelope.
     * @since SAAJ 1.3
     */
    public static final String
                URI_NS_SOAP_1_2_ENVELOPE = "http://www.w3.org/2003/05/soap-envelope";

    /**
     * The namespace identifier for the SOAP 1.1 envelope, All SOAPElements in this
     * namespace are defined by the SOAP 1.1 specification.
     */
    public static final String
        URI_NS_SOAP_ENVELOPE = URI_NS_SOAP_1_1_ENVELOPE;

    /**
     * The namespace identifier for the SOAP 1.1 encoding.
     * An attribute named <code>encodingStyle</code> in the
     * <code>URI_NS_SOAP_ENVELOPE</code> namespace and set to the value
     * <code>URI_NS_SOAP_ENCODING</code> can be added to an element to indicate
     * that it is encoded using the rules in section 5 of the SOAP 1.1
     * specification.
     */
    public static final String
        URI_NS_SOAP_ENCODING = "http://schemas.xmlsoap.org/soap/encoding/";

    /**
     * The namespace identifier for the SOAP 1.2 encoding.
     * @since SAAJ 1.3
     */
    public static final String
        URI_NS_SOAP_1_2_ENCODING = "http://www.w3.org/2003/05/soap-encoding";

    /**
     * The media type  of the <code>Content-Type</code> MIME header in SOAP 1.1.
     * @since SAAJ 1.3
     */
    public static final String
        SOAP_1_1_CONTENT_TYPE = "text/xml";

    /**
     * The media type  of the <code>Content-Type</code> MIME header in SOAP 1.2.
     * @since SAAJ 1.3
     */
    public static final String
        SOAP_1_2_CONTENT_TYPE = "application/soap+xml";

```

```java
    /**
     * The URI identifying the next application processing a SOAP request as the intended
     * actor for a SOAP 1.1 header entry (see section 4.2.2 of the SOAP 1.1 specification).
     * <p>
     * This value can be passed to
     * {@link SOAPHeader#examineMustUnderstandHeaderElements(String)},
     * {@link SOAPHeader#examineHeaderElements(String)} and
     * {@link SOAPHeader#extractHeaderElements(String)}
     */
    public static final String
        URI_SOAP_ACTOR_NEXT = "http://schemas.xmlsoap.org/soap/actor/next";

    /**
     * The URI identifying the next application processing a SOAP request as the intended
     * role for a SOAP 1.2 header entry (see section 2.2 of part 1 of the SOAP 1.2
     * specification).
     * @since SAAJ 1.3
     */
    public static final String
        URI_SOAP_1_2_ROLE_NEXT = URI_NS_SOAP_1_2_ENVELOPE + "/role/next";

    /**
     * The URI specifying the role None in SOAP 1.2.
     * @since SAAJ 1.3
     */
    public static final String
        URI_SOAP_1_2_ROLE_NONE = URI_NS_SOAP_1_2_ENVELOPE + "/role/none";

    /**
     * The URI identifying the ultimate receiver of the SOAP 1.2 message.
     * @since SAAJ 1.3
     */
    public static final String
        URI_SOAP_1_2_ROLE_ULTIMATE_RECEIVER =
            URI_NS_SOAP_1_2_ENVELOPE + "/role/ultimateReceiver";

    /**
     * The default namespace prefix for http://www.w3.org/2003/05/soap-envelope
     * @since SAAJ 1.3
     */
    public static final String SOAP_ENV_PREFIX = "env";

    /**
     * SOAP 1.2 VersionMismatch Fault
     * @since SAAJ 1.3
     */
    public static final QName SOAP_VERSIONMISMATCH_FAULT =
         new QName(URI_NS_SOAP_1_2_ENVELOPE, "VersionMismatch", SOAP_ENV_PREFIX);

    /**
     * SOAP 1.2 MustUnderstand Fault
     * @since SAAJ 1.3
     */
```

```
    public static final QName SOAP_MUSTUNDERSTAND_FAULT =
         new QName(URI_NS_SOAP_1_2_ENVELOPE, "MustUnderstand", SOAP_ENV_PREFIX);

    /**
     * SOAP 1.2 DataEncodingUnknown Fault
     * @since SAAJ 1.3
     */
    public static final QName SOAP_DATAENCODINGUNKNOWN_FAULT =
         new QName(URI_NS_SOAP_1_2_ENVELOPE, "DataEncodingUnknown", SOAP_ENV_PREFIX);

    /**
     * SOAP 1.2 Sender Fault
     * @since SAAJ 1.3
     */
    public static final QName SOAP_SENDER_FAULT =
         new QName(URI_NS_SOAP_1_2_ENVELOPE, "Sender", SOAP_ENV_PREFIX);

    /**
     * SOAP 1.2 Receiver Fault
     * @since SAAJ 1.3
     */
    public static final QName SOAP_RECEIVER_FAULT =
         new QName(URI_NS_SOAP_1_2_ENVELOPE, "Receiver", SOAP_ENV_PREFIX);

}
```

17.17 SOAPElement.java

Secuencia 178: javax/xml/soap/SOAPElement.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```java

package javax.xml.soap;

import java.util.Iterator;

import javax.xml.namespace.QName;

/**
 * An object representing an element of a SOAP message that is allowed but not
 * specifically prescribed by a SOAP specification. This interface serves as the
 * base interface for those objects that are specifically prescribed by a SOAP
 * specification.
 * <p>
 * Methods in this interface that are required to return SAAJ specific objects
 * may "silently" replace nodes in the tree as required to successfully return
 * objects of the correct type. See {@link #getChildElements()} and
 * {@link <a HREF="package-summary.html#package_description">javax.xml.soap<a>}
 * for details.
 */
public interface SOAPElement extends Node, org.w3c.dom.Element {

    /**
     * Creates a new <code>SOAPElement</code> object initialized with the
     * given <code>Name</code> object and adds the new element to this
     * <code>SOAPElement</code> object.
     * <P>
     * This method may be deprecated in a future release of SAAJ in favor of
     * addChildElement(javax.xml.namespace.QName)
     *
     * @param name a <code>Name</code> object with the XML name for the
     *        new element
     *
     * @return the new <code>SOAPElement</code> object that was created
     * @exception SOAPException if there is an error in creating the
     *                          <code>SOAPElement</code> object
     * @see SOAPElement#addChildElement(javax.xml.namespace.QName)
     */
    public SOAPElement addChildElement(Name name) throws SOAPException;

    /**
     * Creates a new <code>SOAPElement</code> object initialized with the given
     * <code>QName</code> object and adds the new element to this <code>SOAPElement</code>
     *  object. The  <i>namespace</i>, <i>localname</i> and <i>prefix</i> of the new
     * <code>SOAPElement</code> are all taken  from the <code>qname</code> argument.
     *
     * @param qname a <code>QName</code> object with the XML name for the
     *        new element
     *
     * @return the new <code>SOAPElement</code> object that was created
     * @exception SOAPException if there is an error in creating the
     *                          <code>SOAPElement</code> object
     * @see SOAPElement#addChildElement(Name)
     * @since SAAJ 1.3
```

```
     */
    public SOAPElement addChildElement(QName qname) throws SOAPException;

    /**
     * Creates a new <code>SOAPElement</code> object initialized with the
     * specified local name and adds the new element to this
     * <code>SOAPElement</code> object.
     * The new  <code>SOAPElement</code> inherits any in-scope default namespace.
     *
     * @param localName a <code>String</code> giving the local name for
     *          the element
     * @return the new <code>SOAPElement</code> object that was created
     * @exception SOAPException if there is an error in creating the
     *                          <code>SOAPElement</code> object
     */
    public SOAPElement addChildElement(String localName) throws SOAPException;

    /**
     * Creates a new <code>SOAPElement</code> object initialized with the
     * specified local name and prefix and adds the new element to this
     * <code>SOAPElement</code> object.
     *
     * @param localName a <code>String</code> giving the local name for
     *        the new element
     * @param prefix a <code>String</code> giving the namespace prefix for
     *        the new element
     *
     * @return the new <code>SOAPElement</code> object that was created
     * @exception SOAPException if the <code>prefix</code> is not valid in the
     *         context of this <code>SOAPElement</code> or  if there is an error in creating the
     *                          <code>SOAPElement</code> object
     */
    public SOAPElement addChildElement(String localName, String prefix)
        throws SOAPException;

    /**
     * Creates a new <code>SOAPElement</code> object initialized with the
     * specified local name, prefix, and URI and adds the new element to this
     * <code>SOAPElement</code> object.
     *
     * @param localName a <code>String</code> giving the local name for
     *        the new element
     * @param prefix a <code>String</code> giving the namespace prefix for
     *        the new element
     * @param uri a <code>String</code> giving the URI of the namespace
     *        to which the new element belongs
     *
     * @return the new <code>SOAPElement</code> object that was created
     * @exception SOAPException if there is an error in creating the
     *                          <code>SOAPElement</code> object
     */
    public SOAPElement addChildElement(String localName, String prefix,
                                       String uri)
```

```
        throws SOAPException;

    /**
     * Add a <code>SOAPElement</code> as a child of this
     * <code>SOAPElement</code> instance. The <code>SOAPElement</code>
     * is expected to be created by a
     * <code>SOAPFactory</code>. Callers should not rely on the
     * element instance being added as is into the XML
     * tree. Implementations could end up copying the content
     * of the <code>SOAPElement</code> passed into an instance of
     * a different <code>SOAPElement</code> implementation. For
     * instance if <code>addChildElement()</code> is called on a
     * <code>SOAPHeader</code>, <code>element</code> will be copied
     * into an instance of a <code>SOAPHeaderElement</code>.
     *
     * <P>The fragment rooted in <code>element</code> is either added
     * as a whole or not at all, if there was an error.
     *
     * <P>The fragment rooted in <code>element</code> cannot contain
     * elements named "Envelope", "Header" or "Body" and in the SOAP
     * namespace. Any namespace prefixes present in the fragment
     * should be fully resolved using appropriate namespace
     * declarations within the fragment itself.
     *
     * @param element the <code>SOAPElement</code> to be added as a
     *                new child
     *
     * @exception SOAPException if there was an error in adding this
     *                          element as a child
     *
     * @return an instance representing the new SOAP element that was
     *         actually added to the tree.
     */
    public SOAPElement addChildElement(SOAPElement element)
        throws SOAPException;

    /**
     * Detaches all children of this <code>SOAPElement</code>.
     * <p>
     * This method is useful for rolling back the construction of partially
     * completed <code>SOAPHeaders</code> and <code>SOAPBodys</code> in
     * preparation for sending a fault when an error condition is detected. It
     * is also useful for recycling portions of a document within a SOAP
     * message.
     *
     * @since SAAJ 1.2
     */
    public abstract void removeContents();

    /**
     * Creates a new <code>Text</code> object initialized with the given
     * <code>String</code> and adds it to this <code>SOAPElement</code> object.
     *
```

```
     * @param text a <code>String</code> object with the textual content to be added
     *
     * @return the <code>SOAPElement</code> object into which
     *         the new <code>Text</code> object was inserted
     * @exception SOAPException if there is an error in creating the
     *                    new <code>Text</code> object or if it is not legal to
     *                      attach it as a child to this
     *                      <code>SOAPElement</code>
     */
    public SOAPElement addTextNode(String text) throws SOAPException;

    /**
     * Adds an attribute with the specified name and value to this
     * <code>SOAPElement</code> object.
     *
     * @param name a <code>Name</code> object with the name of the attribute
     * @param value a <code>String</code> giving the value of the attribute
     * @return the <code>SOAPElement</code> object into which the attribute was
     *         inserted
     *
     * @exception SOAPException if there is an error in creating the
     *                          Attribute, or it is invalid to set
                                an attribute with <code>Name</code>
                                 <code>name</code> on this SOAPElement.
     * @see SOAPElement#addAttribute(javax.xml.namespace.QName, String)
     */
    public SOAPElement addAttribute(Name name, String value)
        throws SOAPException;

    /**
     * Adds an attribute with the specified name and value to this
     * <code>SOAPElement</code> object.
     *
     * @param qname a <code>QName</code> object with the name of the attribute
     * @param value a <code>String</code> giving the value of the attribute
     * @return the <code>SOAPElement</code> object into which the attribute was
     *         inserted
     *
     * @exception SOAPException if there is an error in creating the
     *                          Attribute, or it is invalid to set
                                an attribute with <code>QName</code>
                                <code>qname</code> on this SOAPElement.
     * @see SOAPElement#addAttribute(Name, String)
     * @since SAAJ 1.3
     */
    public SOAPElement addAttribute(QName qname, String value)
        throws SOAPException;

    /**
     * Adds a namespace declaration with the specified prefix and URI to this
     * <code>SOAPElement</code> object.
     *
     * @param prefix a <code>String</code> giving the prefix of the namespace
```

```
     * @param uri a <code>String</code> giving the uri of the namespace
     * @return the <code>SOAPElement</code> object into which this
     *          namespace declaration was inserted.
     *
     * @exception SOAPException if there is an error in creating the
     *                          namespace
     */
    public SOAPElement addNamespaceDeclaration(String prefix, String uri)
        throws SOAPException;

    /**
     * Returns the value of the attribute with the specified name.
     *
     * @param name a <code>Name</code> object with the name of the attribute
     * @return a <code>String</code> giving the value of the specified
     *         attribute, Null if there is no such attribute
     * @see SOAPElement#getAttributeValue(javax.xml.namespace.QName)
     */
    public String getAttributeValue(Name name);

    /**
     * Returns the value of the attribute with the specified qname.
     *
     * @param qname a <code>QName</code> object with the qname of the attribute
     * @return a <code>String</code> giving the value of the specified
     *         attribute, Null if there is no such attribute
     * @see SOAPElement#getAttributeValue(Name)
     * @since SAAJ 1.3
     */
    public String getAttributeValue(QName qname);

    /**
     * Returns an <code>Iterator</code> over all of the attribute
     * <code>Name</code> objects in this
     * <code>SOAPElement</code> object. The iterator can be used to get
     * the attribute names, which can then be passed to the method
     * <code>getAttributeValue</code> to retrieve the value of each
     * attribute.
     *
     * @see SOAPElement#getAllAttributesAsQNames()
     * @return an iterator over the names of the attributes
     */
    public Iterator getAllAttributes();

    /**
     * Returns an <code>Iterator</code> over all of the attributes
     * in this <code>SOAPElement</code>  as <code>QName</code> objects.
     * The iterator can be used to get the attribute QName, which can then
     * be passed to the method <code>getAttributeValue</code> to retrieve
     * the value of each attribute.
     *
     * @return an iterator over the QNames of the attributes
     * @see SOAPElement#getAllAttributes()
```

```
     * @since SAAJ 1.3
     */
    public Iterator getAllAttributesAsQNames();


    /**
     * Returns the URI of the namespace that has the given prefix.
     *
     * @param prefix a <code>String</code> giving the prefix of the namespace
     *        for which to search
     * @return a <code>String</code> with the uri of the namespace that has
     *        the given prefix
     */
    public String getNamespaceURI(String prefix);

    /**
     * Returns an <code>Iterator</code> over the namespace prefix
     * <code>String</code>s declared by this element. The prefixes returned by
     * this iterator can be passed to the method
     * <code>getNamespaceURI</code> to retrieve the URI of each namespace.
     *
     * @return an iterator over the namespace prefixes in this
     *         <code>SOAPElement</code> object
     */
    public Iterator getNamespacePrefixes();

    /**
     * Returns an <code>Iterator</code> over the namespace prefix
     * <code>String</code>s visible to this element. The prefixes returned by
     * this iterator can be passed to the method
     * <code>getNamespaceURI</code> to retrieve the URI of each namespace.
     *
     * @return an iterator over the namespace prefixes are within scope of this
     *         <code>SOAPElement</code> object
     *
     * @since SAAJ 1.2
     */
    public Iterator getVisibleNamespacePrefixes();

    /**
     * Creates a <code>QName</code> whose namespace URI is the one associated
     * with the parameter, <code>prefix</code>, in the context of this
     * <code>SOAPElement</code>. The remaining elements of the new
     * <code>QName</code> are taken directly from the parameters,
     * <code>localName</code> and <code>prefix</code>.
     *
     * @param localName
     *          a <code>String</code> containing the local part of the name.
     * @param prefix
     *          a <code>String</code> containing the prefix for the name.
     *
     * @return a <code>QName</code> with the specified <code>localName</code>
     *          and <code>prefix</code>, and with a namespace that is associated
```

```
     *          with the <code>prefix</code> in the context of this
     *          <code>SOAPElement</code>. This namespace will be the same as
     *          the one that would be returned by
     *          <code>{@link #getNamespaceURI(String)}</code> if it were given
     *          <code>prefix</code> as it's parameter.
     *
     * @exception SOAPException if the <code>QName</code> cannot be created.
     *
     * @since SAAJ 1.3
     */
    public QName createQName(String localName, String prefix)
        throws SOAPException;
    /**
     * Returns the name of this <code>SOAPElement</code> object.
     *
     * @return a <code>Name</code> object with the name of this
     *         <code>SOAPElement</code> object
     */
    public Name getElementName();

    /**
     * Returns the qname of this <code>SOAPElement</code> object.
     *
     * @return a <code>QName</code> object with the qname of this
     *         <code>SOAPElement</code> object
     * @see SOAPElement#getElementName()
     * @since SAAJ 1.3
     */
    public QName getElementQName();

    /**
    * Changes the name of this <code>Element</code> to <code>newName</code> if
    * possible. SOAP Defined elements such as SOAPEnvelope, SOAPHeader, SOAPBody
    * etc. cannot have their names changed using this method. Any attempt to do
    * so will result in a  SOAPException being thrown.
    *<P>
    * Callers should not rely on the element instance being renamed as is.
    * Implementations could end up copying the content of the
    * <code>SOAPElement</code> to a renamed instance.
    *
    * @param newName the new name for the <code>Element</code>.
    *
    * @exception SOAPException if changing the name of this <code>Element</code>
    *                          is not allowed.
    * @return The renamed Node
    *
    * @since SAAJ 1.3
    */
   public SOAPElement setElementQName(QName newName) throws SOAPException;

   /**
     * Removes the attribute with the specified name.
     *
```

```
     * @param name the <code>Name</code> object with the name of the
     *        attribute to be removed
     * @return <code>true</code> if the attribute was
     *         removed successfully; <code>false</code> if it was not
     * @see SOAPElement#removeAttribute(javax.xml.namespace.QName)
     */
    public boolean removeAttribute(Name name);

    /**
     * Removes the attribute with the specified qname.
     *
     * @param qname the <code>QName</code> object with the qname of the
     *        attribute to be removed
     * @return <code>true</code> if the attribute was
     *         removed successfully; <code>false</code> if it was not
     * @see SOAPElement#removeAttribute(Name)
     * @since SAAJ 1.3
     */
    public boolean removeAttribute(QName qname);

    /**
     * Removes the namespace declaration corresponding to the given prefix.
     *
     * @param prefix a <code>String</code> giving the prefix for which
     *        to search
     * @return <code>true</code> if the namespace declaration was
     *         removed successfully; <code>false</code> if it was not
     */
    public boolean removeNamespaceDeclaration(String prefix);

    /**
     * Returns an <code>Iterator</code> over all the immediate child
     * {@link Node}s of this element. This includes <code>javax.xml.soap.Text</code>
     * objects as well as <code>SOAPElement</code> objects.
     * <p>
     * Calling this method may cause child <code>Element</code>,
     * <code>SOAPElement</code> and <code>org.w3c.dom.Text</code> nodes to be
     * replaced by <code>SOAPElement</code>, <code>SOAPHeaderElement</code>,
     * <code>SOAPBodyElement</code> or <code>javax.xml.soap.Text</code> nodes as
     * appropriate for the type of this parent node. As a result the calling
     * application must treat any existing references to these child nodes that
     * have been obtained through DOM APIs as invalid and either discard them or
     * refresh them with the values returned by this <code>Iterator</code>. This
     * behavior can be avoided by calling the equivalent DOM APIs. See
     * {@link <a HREF="package-summary.html#package_description">javax.xml.soap<a>}
     * for more details.
     *
     * @return an iterator with the content of this <code>SOAPElement</code>
     *         object
     */
    public Iterator getChildElements();

    /**
```

```
     * Returns an <code>Iterator</code> over all the immediate child
     * {@link Node}s of this element with the specified name. All of these
     * children will be <code>SOAPElement</code> nodes.
     * <p>
     * Calling this method may cause child <code>Element</code>,
     * <code>SOAPElement</code> and <code>org.w3c.dom.Text</code> nodes to be
     * replaced by <code>SOAPElement</code>, <code>SOAPHeaderElement</code>,
     * <code>SOAPBodyElement</code> or <code>javax.xml.soap.Text</code> nodes as
     * appropriate for the type of this parent node. As a result the calling
     * application must treat any existing references to these child nodes that
     * have been obtained through DOM APIs as invalid and either discard them or
     * refresh them with the values returned by this <code>Iterator</code>. This
     * behavior can be avoided by calling the equivalent DOM APIs. See
     * {@link <a HREF="package-summary.html#package_description">javax.xml.soap<a>}
     * for more details.
     *
     * @param name a <code>Name</code> object with the name of the child
     *        elements to be returned
     *
     * @return an <code>Iterator</code> object over all the elements
     *         in this <code>SOAPElement</code> object with the
     *         specified name
     * @see SOAPElement#getChildElements(javax.xml.namespace.QName)
     */
    public Iterator getChildElements(Name name);

    /**
     * Returns an <code>Iterator</code> over all the immediate child
     * {@link Node}s of this element with the specified qname. All of these
     * children will be <code>SOAPElement</code> nodes.
     * <p>
     * Calling this method may cause child <code>Element</code>,
     * <code>SOAPElement</code> and <code>org.w3c.dom.Text</code> nodes to be
     * replaced by <code>SOAPElement</code>, <code>SOAPHeaderElement</code>,
     * <code>SOAPBodyElement</code> or <code>javax.xml.soap.Text</code> nodes as
     * appropriate for the type of this parent node. As a result the calling
     * application must treat any existing references to these child nodes that
     * have been obtained through DOM APIs as invalid and either discard them or
     * refresh them with the values returned by this <code>Iterator</code>. This
     * behavior can be avoided by calling the equivalent DOM APIs. See
     * {@link <a HREF="package-summary.html#package_description">javax.xml.soap<a>}
     * for more details.
     *
     * @param qname a <code>QName</code> object with the qname of the child
     *        elements to be returned
     *
     * @return an <code>Iterator</code> object over all the elements
     *         in this <code>SOAPElement</code> object with the
     *         specified qname
     * @see SOAPElement#getChildElements(Name)
     * @since SAAJ 1.3
     */
    public Iterator getChildElements(QName qname);
```

```

    /**
     * Sets the encoding style for this <code>SOAPElement</code> object
     * to one specified.
     *
     * @param encodingStyle a <code>String</code> giving the encoding style
     *
     * @exception IllegalArgumentException if there was a problem in the
     *            encoding style being set.
     * @exception SOAPException if setting the encodingStyle is invalid for this SOAPElement.
     * @see #getEncodingStyle
     */
    public void setEncodingStyle(String encodingStyle)
        throws SOAPException;
    /**
     * Returns the encoding style for this <code>SOAPElement</code> object.
     *
     * @return a <code>String</code> giving the encoding style
     *
     * @see #setEncodingStyle
     */
    public String getEncodingStyle();
}
```

17.18 SOAPElementFactory.java

Secuencia 179: javax/xml/soap/SOAPElementFactory.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;
```

```java

/**
 * <code>SOAPElementFactory</code> is a factory for XML fragments that
 * will eventually end up in the SOAP part. These fragments
 * can be inserted as children of the <code>SOAPHeader</code> or
 * <code>SOAPBody</code> or <code>SOAPEnvelope</code>.
 *
 * <p>Elements created using this factory do not have the properties
 * of an element that lives inside a SOAP header document. These
 * elements are copied into the XML document tree when they are
 * inserted.
 * @deprecated - Use <code>javax.xml.soap.SOAPFactory</code> for creating SOAPElements.
 * @see javax.xml.soap.SOAPFactory
 */
public class SOAPElementFactory {

    private SOAPFactory soapFactory;

    private SOAPElementFactory(SOAPFactory soapFactory) {
        this.soapFactory = soapFactory;
    }

    /**
     * Create a <code>SOAPElement</code> object initialized with the
     * given <code>Name</code> object.
     *
     * @param name a <code>Name</code> object with the XML name for
     *             the new element
     *
     * @return the new <code>SOAPElement</code> object that was
     *         created
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     *
     * @deprecated Use
     * javax.xml.soap.SOAPFactory.createElement(javax.xml.soap.Name)
     * instead
     *
     * @see javax.xml.soap.SOAPFactory#createElement(javax.xml.soap.Name)
     * @see javax.xml.soap.SOAPFactory#createElement(javax.xml.namespace.QName)
     */
    public SOAPElement create(Name name) throws SOAPException {
        return soapFactory.createElement(name);
    }

    /**
     * Create a <code>SOAPElement</code> object initialized with the
     * given local name.
     *
     * @param localName a <code>String</code> giving the local name for
     *             the new element
     *
```

```
     * @return the new <code>SOAPElement</code> object that was
     *         created
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     *
     * @deprecated Use
     * javax.xml.soap.SOAPFactory.createElement(String localName) instead
     *
     * @see javax.xml.soap.SOAPFactory#createElement(java.lang.String)
     */
    public SOAPElement create(String localName) throws SOAPException {
        return soapFactory.createElement(localName);
    }

    /**
     * Create a new <code>SOAPElement</code> object with the given
     * local name, prefix and uri.
     *
     * @param localName a <code>String</code> giving the local name
     *                  for the new element
     * @param prefix the prefix for this <code>SOAPElement</code>
     * @param uri a <code>String</code> giving the URI of the
     *            namespace to which the new element belongs
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     *
     * @deprecated Use
     * javax.xml.soap.SOAPFactory.createElement(String localName,
     *                      String prefix,
     *                      String uri)
     * instead
     *
     * @see javax.xml.soap.SOAPFactory#createElement(java.lang.String, java.lang.String, java.lang.
       String)
     */
    public SOAPElement create(String localName, String prefix, String uri)
        throws SOAPException {
        return soapFactory.createElement(localName, prefix, uri);
    }

    /**
     * Creates a new instance of <code>SOAPElementFactory</code>.
     *
     * @return a new instance of a <code>SOAPElementFactory</code>
     *
     * @exception SOAPException if there was an error creating the
     *            default <code>SOAPElementFactory</code>
     */
    public static SOAPElementFactory newInstance() throws SOAPException {
        try {
            return new SOAPElementFactory(SOAPFactory.newInstance());
```

```
        } catch (Exception ex) {
            throw new SOAPException(
                "Unable to create SOAP Element Factory: " + ex.getMessage());
        }
    }
}
```

17.19 SOAPEnvelope.java

Secuencia 180: javax/xml/soap/SOAPEnvelope.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;


/**
 * The container for the SOAPHeader and SOAPBody portions of a
 * <code>SOAPPart</code> object. By default, a <code>SOAPMessage</code>
 * object is created with a <code>SOAPPart</code> object that has a
 * <code>SOAPEnvelope</code> object. The <code>SOAPEnvelope</code> object
 * by default has an empty <code>SOAPBody</code> object and an empty
 * <code>SOAPHeader</code> object.  The <code>SOAPBody</code> object is
 * required, and the <code>SOAPHeader</code> object, though
 * optional, is used in the majority of cases. If the
 * <code>SOAPHeader</code> object is not needed, it can be deleted,
 * which is shown later.
 * <P>
 * A client can access the <code>SOAPHeader</code> and <code>SOAPBody</code>
 * objects by calling the methods <code>SOAPEnvelope.getHeader</code> and
 * <code>SOAPEnvelope.getBody</code>. The
```

```
* following  lines of code use these two methods after starting with
* the <code>SOAPMessage</code>
* object <i>message</i> to get the <code>SOAPPart</code> object <i>sp</i>,
* which is then used to get the <code>SOAPEnvelope</code> object <i>se</i>.
*
* <PRE>
*     SOAPPart sp = message.getSOAPPart();
*     SOAPEnvelope se = sp.getEnvelope();
*     SOAPHeader sh = se.getHeader();
*     SOAPBody sb = se.getBody();
* </PRE>
* <P>
* It is possible to change the body or header of a <code>SOAPEnvelope</code>
* object by retrieving the current one, deleting it, and then adding
* a new body or header. The <code>javax.xml.soap.Node</code> method
* <code>deleteNode</code> deletes the XML element (node) on which it is
* called.  For example, the following line of code deletes the
* <code>SOAPBody</code> object that is retrieved by the method <code>getBody</code>.
* <PRE>
*      se.getBody().detachNode();
* </PRE>
* To create a <code>SOAPHeader</code> object to replace the one that was removed,
* a client uses
* the method <code>SOAPEnvelope.addHeader</code>, which creates a new header and
* adds it to the <code>SOAPEnvelope</code> object. Similarly, the method
* <code>addBody</code> creates a new <code>SOAPBody</code> object and adds
* it to the <code>SOAPEnvelope</code> object. The following code fragment
* retrieves the current header, removes it, and adds a new one. Then
* it retrieves the current body, removes it, and adds a new one.
*
* <PRE>
*     SOAPPart sp = message.getSOAPPart();
*     SOAPEnvelope se = sp.getEnvelope();
*     se.getHeader().detachNode();
*     SOAPHeader sh = se.addHeader();
*     se.getBody().detachNode();
*     SOAPBody sb = se.addBody();
* </PRE>
* It is an error to add a <code>SOAPBody</code> or <code>SOAPHeader</code>
* object if one already exists.
* <P>
* The <code>SOAPEnvelope</code> interface provides three methods for creating
* <code>Name</code> objects. One method creates <code>Name</code> objects with
* a local name, a namespace prefix, and a namesapce URI. The second method creates
* <code>Name</code> objects with a local name and a namespace prefix, and the third
* creates <code>Name</code> objects with just a local name.  The following line of
* code, in which <i>se</i> is a <code>SOAPEnvelope</code> object, creates a new
* <code>Name</code> object with all three.
* <PRE>
*     Name name = se.createName("GetLastTradePrice", "WOMBAT",
*                               "http://www.wombat.org/trader");
* </PRE>
*/
```

```java
public interface SOAPEnvelope extends SOAPElement {

    /**
     * Creates a new <code>Name</code> object initialized with the
     * given local name, namespace prefix, and namespace URI.
     * <P>
     * This factory method creates <code>Name</code> objects for use in
     * the SOAP/XML document.
     *
     * @param localName a <code>String</code> giving the local name
     * @param prefix a <code>String</code> giving the prefix of the namespace
     * @param uri a <code>String</code> giving the URI of the namespace
     * @return a <code>Name</code> object initialized with the given
     *         local name, namespace prefix, and namespace URI
     * @throws SOAPException if there is a SOAP error
     */
    public abstract Name createName(String localName, String prefix,
                                    String uri)
        throws SOAPException;

    /**
     * Creates a new <code>Name</code> object initialized with the
     * given local name.
     * <P>
     * This factory method creates <code>Name</code> objects for use in
     * the SOAP/XML document.
     *
     * @param localName a <code>String</code> giving the local name
     * @return a <code>Name</code> object initialized with the given
     *         local name
     * @throws SOAPException if there is a SOAP error
     */
    public abstract Name createName(String localName)
        throws SOAPException;

    /**
     * Returns the <code>SOAPHeader</code> object for
     * this <code>SOAPEnvelope</code> object.
     * <P>
     * A new <code>SOAPMessage</code> object is by default created with a
     * <code>SOAPEnvelope</code> object that contains an empty
     * <code>SOAPHeader</code> object.  As a result, the method
     * <code>getHeader</code> will always return a <code>SOAPHeader</code>
     * object unless the header has been removed and a new one has not
     * been added.
     *
     * @return the <code>SOAPHeader</code> object or <code>null</code> if
     *         there is none
     * @exception SOAPException if there is a problem obtaining the
     *            <code>SOAPHeader</code> object
     */
    public SOAPHeader getHeader() throws SOAPException;

```

```
    /**
     * Returns the <code>SOAPBody</code> object associated with this
     * <code>SOAPEnvelope</code> object.
     * <P>
     * A new <code>SOAPMessage</code> object is by default created with a
     * <code>SOAPEnvelope</code> object that contains an empty
     * <code>SOAPBody</code> object.  As a result, the method
     * <code>getBody</code> will always return a <code>SOAPBody</code>
     * object unless the body has been removed and a new one has not
     * been added.
     *
     * @return the <code>SOAPBody</code> object for this
     *         <code>SOAPEnvelope</code> object or <code>null</code>
     *         if there is none
     * @exception SOAPException if there is a problem obtaining the
     *            <code>SOAPBody</code> object
     */
    public SOAPBody getBody() throws SOAPException;
    /**
     * Creates a <code>SOAPHeader</code> object and sets it as the
     * <code>SOAPHeader</code> object for this <code>SOAPEnvelope</code>
     * object.
     * <P>
     * It is illegal to add a header when the envelope already
     * contains a header.  Therefore, this method should be called
     * only after the existing header has been removed.
     *
     * @return the new <code>SOAPHeader</code> object
     *
     * @exception SOAPException if this
     *            <code>SOAPEnvelope</code> object already contains a
     *            valid <code>SOAPHeader</code> object
     */
    public SOAPHeader addHeader() throws SOAPException;
    /**
     * Creates a <code>SOAPBody</code> object and sets it as the
     * <code>SOAPBody</code> object for this <code>SOAPEnvelope</code>
     * object.
     * <P>
     * It is illegal to add a body when the envelope already
     * contains a body. Therefore, this method should be called
     * only after the existing body has been removed.
     *
     * @return the new <code>SOAPBody</code> object
     *
     * @exception SOAPException if this
     *            <code>SOAPEnvelope</code> object already contains a
     *            valid <code>SOAPBody</code> object
     */
    public SOAPBody addBody() throws SOAPException;
}
```

17.20 SOAPException.java

Secuencia 181: javax/xml/soap/SOAPException.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * An exception that signals that a SOAP exception has occurred. A
 * <code>SOAPException</code> object may contain a <code>String</code>
 * that gives the reason for the exception, an embedded
 * <code>Throwable</code> object, or both. This class provides methods
 * for retrieving reason messages and for retrieving the embedded
 * <code>Throwable</code> object.
 *
 * <P> Typical reasons for throwing a <code>SOAPException</code>
 * object are problems such as difficulty setting a header, not being
 * able to send a message, and not being able to get a connection with
 * the provider.  Reasons for embedding a <code>Throwable</code>
 * object include problems such as input/output errors or a parsing
 * problem, such as an error in parsing a header.
 */
public class SOAPException extends Exception {
    private Throwable cause;

    /**
     * Constructs a <code>SOAPException</code> object with no
     * reason or embedded <code>Throwable</code> object.
     */
    public SOAPException() {
```

```
        super();
        this.cause = null;
    }

    /**
     * Constructs a <code>SOAPException</code> object with the given
     * <code>String</code> as the reason for the exception being thrown.
     *
     * @param reason a description of what caused the exception
     */
    public SOAPException(String reason) {
        super(reason);
        this.cause = null;
    }

    /**
     * Constructs a <code>SOAPException</code> object with the given
     * <code>String</code> as the reason for the exception being thrown
     * and the given <code>Throwable</code> object as an embedded
     * exception.
     *
     * @param reason a description of what caused the exception
     * @param cause a <code>Throwable</code> object that is to
     *        be embedded in this <code>SOAPException</code> object
     */
    public SOAPException(String reason, Throwable cause) {
        super(reason);
        initCause(cause);
    }

    /**
     * Constructs a <code>SOAPException</code> object initialized
     * with the given <code>Throwable</code> object.
     */
    public SOAPException(Throwable cause) {
        super(cause.toString());
        initCause(cause);
    }

    /**
     * Returns the detail message for this <code>SOAPException</code>
     * object.
     * <P>
     * If there is an embedded <code>Throwable</code> object, and if the
     * <code>SOAPException</code> object has no detail message of its
     * own, this method will return the detail message from the embedded
     * <code>Throwable</code> object.
     *
     * @return the error or warning message for this
     *         <code>SOAPException</code> or, if it has none, the
     *         message of the embedded <code>Throwable</code> object,
     *         if there is one
     */
```

```java
    public String getMessage() {
        String message = super.getMessage();
        if (message == null && cause != null) {
            return cause.getMessage();
        } else {
            return message;
        }
    }

    /**
     * Returns the <code>Throwable</code> object embedded in this
     * <code>SOAPException</code> if there is one. Otherwise, this method
     * returns <code>null</code>.
     *
     * @return the embedded <code>Throwable</code> object or <code>null</code>
     *         if there is none
     */

    public Throwable getCause() {
        return cause;
    }

    /**
     * Initializes the <code>cause</code> field of this <code>SOAPException</code>
     * object with the given <code>Throwable</code> object.
     * <P>
     * This method can be called at most once.  It is generally called from
     * within the constructor or immediately after the constructor has
     * returned a new <code>SOAPException</code> object.
     * If this <code>SOAPException</code> object was created with the
     * constructor {@link #SOAPException(Throwable)} or
     * {@link #SOAPException(String,Throwable)}, meaning that its
     * <code>cause</code> field already has a value, this method cannot be
     * called even once.
     *
     * @param  cause the <code>Throwable</code> object that caused this
     *         <code>SOAPException</code> object to be thrown.  The value of this
     *         parameter is saved for later retrieval by the
     *         {@link #getCause()} method.  A <tt>null</tt> value is
     *         permitted and indicates that the cause is nonexistent or
     *         unknown.
     * @return  a reference to this <code>SOAPException</code> instance
     * @throws IllegalArgumentException if <code>cause</code> is this
     *         <code>Throwable</code> object.  (A <code>Throwable</code> object
     *         cannot be its own cause.)
     * @throws IllegalStateException if the cause for this <code>SOAPException</code> object
     *         has already been initialized
     */
    public synchronized Throwable initCause(Throwable cause) {
        if (this.cause != null) {
            throw new IllegalStateException("Can't override cause");
        }
        if (cause == this) {
```

```
            throw new IllegalArgumentException("Self-causation not permitted");
        }
        this.cause = cause;

        return this;
    }
}
```

17.21 SOAPFactory.java

Secuencia 182: javax/xml/soap/SOAPFactory.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import javax.xml.namespace.QName;

import org.w3c.dom.Element;

/**
 * <code>SOAPFactory</code> is a factory for creating various objects
 * that exist in the SOAP XML tree.

 * <code>SOAPFactory</code> can be
 * used to create XML fragments that will eventually end up in the
 * SOAP part. These fragments can be inserted as children of the
 * {@link SOAPHeaderElement} or {@link SOAPBodyElement} or
 * {@link SOAPEnvelope} or other {@link SOAPElement} objects.
 *
 * <code>SOAPFactory</code> also has methods to create
```

```
 * <code>javax.xml.soap.Detail</code> objects as well as
 * <code>java.xml.soap.Name</code> objects.
 *
 */
public abstract class SOAPFactory {

    /**
     * A constant representing the property used to lookup the name of
     * a <code>SOAPFactory</code> implementation class.
     */
    static private final String SOAP_FACTORY_PROPERTY =
        "javax.xml.soap.SOAPFactory";

    /**
     * Class name of default <code>SOAPFactory</code> implementation.
     */
    static final String DEFAULT_SOAP_FACTORY
        = "com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPFactory1_1Impl";

    /**
     * Creates a <code>SOAPElement</code> object from an existing DOM
     * <code>Element</code>. If the DOM <code>Element</code> that is passed in
     * as an argument is already a <code>SOAPElement</code> then this method
     * must return it unmodified without any further work. Otherwise, a new
     * <code>SOAPElement</code> is created and a deep copy is made of the
     * <code>domElement</code> argument. The concrete type of the return value
     * will depend on the name of the <code>domElement</code> argument. If any
     * part of the tree rooted in <code>domElement</code> violates SOAP rules, a
     * <code>SOAPException</code> will be thrown.
     *
     * @param domElement - the <code>Element</code> to be copied.
     *
     * @return a new <code>SOAPElement</code> that is a copy of <code>domElement</code>.
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     *
     * @since SAAJ 1.3
     */
    public SOAPElement createElement(Element domElement) throws SOAPException {
        throw new UnsupportedOperationException("createElement(org.w3c.dom.Element) must be
            overridden by all subclasses of SOAPFactory.");
    }

    /**
     * Creates a <code>SOAPElement</code> object initialized with the
     * given <code>Name</code> object. The concrete type of the return value
     * will depend on the name given to the new <code>SOAPElement</code>. For
     * instance, a new <code>SOAPElement</code> with the name
     * "{http://www.w3.org/2003/05/soap-envelope}Envelope" would cause a
     * <code>SOAPEnvelope</code> that supports SOAP 1.2 behavior to be created.
     *
     * @param name a <code>Name</code> object with the XML name for
```

```
     *             the new element
     *
     * @return the new <code>SOAPElement</code> object that was
     *         created
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     * @see SOAPFactory#createElement(javax.xml.namespace.QName)
     */
    public abstract SOAPElement createElement(Name name) throws SOAPException;

    /**
     * Creates a <code>SOAPElement</code> object initialized with the
     * given <code>QName</code> object. The concrete type of the return value
     * will depend on the name given to the new <code>SOAPElement</code>. For
     * instance, a new <code>SOAPElement</code> with the name
     * "{http://www.w3.org/2003/05/soap-envelope}Envelope" would cause a
     * <code>SOAPEnvelope</code> that supports SOAP 1.2 behavior to be created.
     *
     * @param qname a <code>QName</code> object with the XML name for
     *             the new element
     *
     * @return the new <code>SOAPElement</code> object that was
     *         created
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     * @see SOAPFactory#createElement(Name)
     * @since SAAJ 1.3
     */
    public  SOAPElement createElement(QName qname) throws SOAPException {
        throw new UnsupportedOperationException("createElement(QName) must be overridden by all
            subclasses of SOAPFactory.");
    }

    /**
     * Creates a <code>SOAPElement</code> object initialized with the
     * given local name.
     *
     * @param localName a <code>String</code> giving the local name for
     *             the new element
     *
     * @return the new <code>SOAPElement</code> object that was
     *         created
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     */
    public abstract SOAPElement createElement(String localName)
        throws SOAPException;


    /**
```

```
     * Creates a new <code>SOAPElement</code> object with the given
     * local name, prefix and uri. The concrete type of the return value
     * will depend on the name given to the new <code>SOAPElement</code>. For
     * instance, a new <code>SOAPElement</code> with the name
     * "{http://www.w3.org/2003/05/soap-envelope}Envelope" would cause a
     * <code>SOAPEnvelope</code> that supports SOAP 1.2 behavior to be created.
     *
     * @param localName a <code>String</code> giving the local name
     *                  for the new element
     * @param prefix the prefix for this <code>SOAPElement</code>
     * @param uri a <code>String</code> giving the URI of the
     *            namespace to which the new element belongs
     *
     * @exception SOAPException if there is an error in creating the
     *            <code>SOAPElement</code> object
     */
    public abstract SOAPElement createElement(
        String localName,
        String prefix,
        String uri)
        throws SOAPException;

    /**
     * Creates a new <code>Detail</code> object which serves as a container
     * for <code>DetailEntry</code> objects.
     * <P>
     * This factory method creates <code>Detail</code> objects for use in
     * situations where it is not practical to use the <code>SOAPFault</code>
     * abstraction.
     *
     * @return a <code>Detail</code> object
     * @throws SOAPException if there is a SOAP error
     * @throws UnsupportedOperationException if the protocol specified
     *         for the SOAPFactory was <code>DYNAMIC_SOAP_PROTOCOL</code>
     */
    public abstract Detail createDetail() throws SOAPException;

    /**
     *Creates a new <code>SOAPFault</code> object initialized with the given <code>reasonText</code>
     *  and <code>faultCode</code>
     *@param reasonText the ReasonText/FaultString for the fault
     *@param faultCode the FaultCode for the fault
     *@return a <code>SOAPFault</code> object
     *@throws SOAPException if there is a SOAP error
     *@since SAAJ 1.3
     */
    public abstract SOAPFault createFault(String reasonText, QName faultCode) throws SOAPException;

    /**
     *Creates a new default <code>SOAPFault</code> object
     *@return a <code>SOAPFault</code> object
     *@throws SOAPException if there is a SOAP error
```

```
     *@since SAAJ 1.3
     */
    public abstract SOAPFault createFault() throws SOAPException;

    /**
     * Creates a new <code>Name</code> object initialized with the
     * given local name, namespace prefix, and namespace URI.
     * <P>
     * This factory method creates <code>Name</code> objects for use in
     * situations where it is not practical to use the <code>SOAPEnvelope</code>
     * abstraction.
     *
     * @param localName a <code>String</code> giving the local name
     * @param prefix a <code>String</code> giving the prefix of the namespace
     * @param uri a <code>String</code> giving the URI of the namespace
     * @return a <code>Name</code> object initialized with the given
     *         local name, namespace prefix, and namespace URI
     * @throws SOAPException if there is a SOAP error
     */
    public abstract Name createName(
        String localName,
        String prefix,
        String uri)
        throws SOAPException;

    /**
     * Creates a new <code>Name</code> object initialized with the
     * given local name.
     * <P>
     * This factory method creates <code>Name</code> objects for use in
     * situations where it is not practical to use the <code>SOAPEnvelope</code>
     * abstraction.
     *
     * @param localName a <code>String</code> giving the local name
     * @return a <code>Name</code> object initialized with the given
     *         local name
     * @throws SOAPException if there is a SOAP error
     */
    public abstract Name createName(String localName) throws SOAPException;

    /**
     * Creates a new <code>SOAPFactory</code> object that is an instance of
     * the default implementation (SOAP 1.1),
     *
     * This method uses the following ordered lookup procedure to determine the SOAPFactory
         implementation class to load:
     * <UL>
     *  <LI> Use the javax.xml.soap.SOAPFactory system property.
     *  <LI> Use the properties file "lib/jaxm.properties" in the JRE directory. This configuration
          file is in standard
     * java.util.Properties format and contains the fully qualified name of the implementation
         class with the key being the
     * system property defined above.
```

```
     *  <LI> Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API
     * will look for a classname in the file META-INF/services/javax.xml.soap.SOAPFactory in jars available to the runtime.
     *  <LI> Use the SAAJMetaFactory instance to locate the SOAPFactory implementation class.
     * </UL>
     *
     * @return a new instance of a <code>SOAPFactory</code>
     *
     * @exception SOAPException if there was an error creating the
     *            default <code>SOAPFactory</code>
     * @see SAAJMetaFactory
     */
    public static SOAPFactory newInstance()
        throws SOAPException
    {
        try {
            SOAPFactory factory = (SOAPFactory) FactoryFinder.find(
                    SOAP_FACTORY_PROPERTY, DEFAULT_SOAP_FACTORY, false);
            if (factory != null)
                return factory;
            return newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);
        } catch (Exception ex) {
            throw new SOAPException(
                "Unable to create SOAP Factory: " + ex.getMessage());
        }

    }

    /**
     * Creates a new <code>SOAPFactory</code> object that is an instance of
     * the specified implementation, this method uses the SAAJMetaFactory to
     * locate the implementation class and create the SOAPFactory instance.
     *
     * @return a new instance of a <code>SOAPFactory</code>
     *
     * @param protocol  a string constant representing the protocol of the
     *                  specified SOAP factory implementation. May be
     *                  either <code>DYNAMIC_SOAP_PROTOCOL</code>,
     *                  <code>DEFAULT_SOAP_PROTOCOL</code> (which is the same
     *                  as) <code>SOAP_1_1_PROTOCOL</code>, or
     *                  <code>SOAP_1_2_PROTOCOL</code>.
     *
     * @exception SOAPException if there was an error creating the
     *            specified <code>SOAPFactory</code>
     * @see SAAJMetaFactory
     * @since SAAJ 1.3
     */
    public static SOAPFactory newInstance(String protocol)
        throws SOAPException {
            return SAAJMetaFactory.getInstance().newSOAPFactory(protocol);
    }
}
```

17.22 SOAPFault.java

Secuencia 183: javax/xml/soap/SOAPFault.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.util.Iterator;
import java.util.Locale;

import javax.xml.namespace.QName;

/**
 * An element in the <code>SOAPBody</code> object that contains
 * error and/or status information. This information may relate to
 * errors in the <code>SOAPMessage</code> object or to problems
 * that are not related to the content in the message itself. Problems
 * not related to the message itself are generally errors in
 * processing, such as the inability to communicate with an upstream
 * server.
 * <P>
 * Depending on the <code>protocol</code> specified while creating the
 * <code>MessageFactory</code> instance,  a <code>SOAPFault</code> has
 * sub-elements as defined in the SOAP 1.1/SOAP 1.2 specification.
 */
public interface SOAPFault extends SOAPBodyElement {

    /**
     * Sets this <code>SOAPFault</code> object with the given fault code.
     *
```

```
     * <P> Fault codes, which give information about the fault, are defined
     * in the SOAP 1.1 specification. A fault code is mandatory and must
     * be of type <code>Name</code>. This method provides a convenient
     * way to set a fault code. For example,
     *
     * <PRE>
     * SOAPEnvelope se = ...;
     * // Create a qualified name in the SOAP namespace with a localName
     * // of "Client". Note that prefix parameter is optional and is null
     * // here which causes the implementation to use an appropriate prefix.
     * Name qname = se.createName("Client", null,
     *                            SOAPConstants.URI_NS_SOAP_ENVELOPE);
     * SOAPFault fault = ...;
     * fault.setFaultCode(qname);
     * </PRE>
     * It is preferable to use this method over {@link #setFaultCode(String)}.
     *
     * @param faultCodeQName a <code>Name</code> object giving the fault
     * code to be set. It must be namespace qualified.
     * @see #getFaultCodeAsName
     *
     * @exception SOAPException if there was an error in adding the
     *            <i>faultcode</i> element to the underlying XML tree.
     *
     * @since SAAJ 1.2
     */
    public void setFaultCode(Name faultCodeQName) throws SOAPException;

    /**
     * Sets this <code>SOAPFault</code> object with the given fault code.
     *
     * It is preferable to use this method over {@link #setFaultCode(Name)}.
     *
     * @param faultCodeQName a <code>QName</code> object giving the fault
     * code to be set. It must be namespace qualified.
     * @see #getFaultCodeAsQName
     *
     * @exception SOAPException if there was an error in adding the
     *            <code>faultcode</code> element to the underlying XML tree.
     *
     * @see #setFaultCode(Name)
     * @see #getFaultCodeAsQName()
     *
     * @since SAAJ 1.3
     */
    public void setFaultCode(QName faultCodeQName) throws SOAPException;

    /**
     * Sets this <code>SOAPFault</code> object with the give fault code.
     * <P>
     * Fault codes, which given information about the fault, are defined in
     * the SOAP 1.1 specification. This element is mandatory in SOAP 1.1.
     * Because the fault code is required to be a QName it is preferable to
```

```
     * use the {@link #setFaultCode(Name)} form of this method.
     *
     * @param faultCode a <code>String</code> giving the fault code to be set.
     *         It must be of the form "prefix:localName" where the prefix has
     *         been defined in a namespace declaration.
     * @see #setFaultCode(Name)
     * @see #getFaultCode
     * @see SOAPElement#addNamespaceDeclaration
     *
     * @exception SOAPException if there was an error in adding the
     *            <code>faultCode</code> to the underlying XML tree.
     */
    public void setFaultCode(String faultCode) throws SOAPException;

    /**
     * Gets the mandatory SOAP 1.1 fault code for this
     * <code>SOAPFault</code> object as a SAAJ <code>Name</code> object.
     * The SOAP 1.1 specification requires the value of the "faultcode"
     * element to be of type QName. This method returns the content of the
     * element as a QName in the form of a SAAJ Name object. This method
     * should be used instead of the <code>getFaultCode</code> method since
     * it allows applications to easily access the namespace name without
     * additional parsing.
     *
     * @return a <code>Name</code> representing the faultcode
     * @see #setFaultCode(Name)
     *
     * @since SAAJ 1.2
     */
    public Name getFaultCodeAsName();


    /**
     * Gets the fault code for this
     * <code>SOAPFault</code> object as a <code>QName</code> object.
     *
     * @return a <code>QName</code> representing the faultcode
     *
     * @see #setFaultCode(QName)
     *
     * @since SAAJ 1.3
     */
    public QName getFaultCodeAsQName();

    /**
     * Gets the Subcodes for this <code>SOAPFault</code> as an iterator over
     * <code>QNames</code>.
     *
     * @return an <code>Iterator</code> that accesses a sequence of
     *      <code>QNames</code>. This <code>Iterator</code> should not support
     *      the optional <code>remove</code> method. The order in which the
     *      Subcodes are returned reflects the hierarchy of Subcodes present
     *      in the fault from top to bottom.
```

```
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Subcode.
     *
     * @since SAAJ 1.3
     */
    public Iterator getFaultSubcodes();

    /**
     * Removes any Subcodes that may be contained by this
     * <code>SOAPFault</code>. Subsequent calls to
     * <code>getFaultSubcodes</code> will return an empty iterator until a call
     * to <code>appendFaultSubcode</code> is made.
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Subcode.
     *
     * @since SAAJ 1.3
     */
    public void removeAllFaultSubcodes();

    /**
     * Adds a Subcode to the end of the sequence of Subcodes contained by this
     * <code>SOAPFault</code>. Subcodes, which were introduced in SOAP 1.2, are
     * represented by a recursive sequence of subelements rooted in the
     * mandatory Code subelement of a SOAP Fault.
     *
     * @param subcode a QName containing the Value of the Subcode.
     *
     * @exception SOAPException if there was an error in setting the Subcode
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Subcode.
     *
     * @since SAAJ 1.3
     */
    public void appendFaultSubcode(QName subcode) throws SOAPException;

    /**
     * Gets the fault code for this <code>SOAPFault</code> object.
     *
     * @return a <code>String</code> with the fault code
     * @see #getFaultCodeAsName
     * @see #setFaultCode
     */
    public String getFaultCode();

    /**
     * Sets this <code>SOAPFault</code> object with the given fault actor.
     * <P>
     * The fault actor is the recipient in the message path who caused the
     * fault to happen.
     * <P>
     * If this <code>SOAPFault</code> supports SOAP 1.2 then this call is
```

```
     * equivalent to {@link #setFaultRole(String)}
     *
     * @param faultActor a <code>String</code> identifying the actor that
     *        caused this <code>SOAPFault</code> object
     * @see #getFaultActor
     *
     * @exception SOAPException if there was an error in adding the
     *            <code>faultActor</code> to the underlying XML tree.
     */
    public void setFaultActor(String faultActor) throws SOAPException;

    /**
     * Gets the fault actor for this <code>SOAPFault</code> object.
     * <P>
     * If this <code>SOAPFault</code> supports SOAP 1.2 then this call is
     * equivalent to {@link #getFaultRole()}
     *
     * @return a <code>String</code> giving the actor in the message path
     *         that caused this <code>SOAPFault</code> object
     * @see #setFaultActor
     */
    public String getFaultActor();

    /**
     * Sets the fault string for this <code>SOAPFault</code> object
     * to the given string.
     * <P>
     * If this
     * <code>SOAPFault</code> is part of a message that supports SOAP 1.2 then
     * this call is equivalent to:
     * <pre>
     *      addFaultReasonText(faultString, Locale.getDefault());
     * </pre>
     *
     * @param faultString a <code>String</code> giving an explanation of
     *        the fault
     * @see #getFaultString
     *
     * @exception SOAPException if there was an error in adding the
     *            <code>faultString</code> to the underlying XML tree.
     */
    public void setFaultString(String faultString) throws SOAPException;

    /**
     * Sets the fault string for this <code>SOAPFault</code> object
     * to the given string and localized to the given locale.
     * <P>
     * If this
     * <code>SOAPFault</code> is part of a message that supports SOAP 1.2 then
     * this call is equivalent to:
     * <pre>
     *      addFaultReasonText(faultString, locale);
     * </pre>
```

```
     *
     * @param faultString a <code>String</code> giving an explanation of
     *         the fault
     * @param locale a {@link java.util.Locale Locale} object indicating
     *         the native language of the <code>faultString</code>
     * @see #getFaultString
     *
     * @exception SOAPException if there was an error in adding the
     *            <code>faultString</code> to the underlying XML tree.
     *
     * @since SAAJ 1.2
     */
    public void setFaultString(String faultString, Locale locale)
        throws SOAPException;

    /**
     * Gets the fault string for this <code>SOAPFault</code> object.
     * <P>
     * If this
     * <code>SOAPFault</code> is part of a message that supports SOAP 1.2 then
     * this call is equivalent to:
     * <pre>
     *    String reason = null;
     *    try {
     *        reason = (String) getFaultReasonTexts().next();
     *    } catch (SOAPException e) {}
     *    return reason;
     * </pre>
     *
     * @return a <code>String</code> giving an explanation of
     *        the fault
     * @see #setFaultString(String)
     * @see #setFaultString(String, Locale)
     */
    public String getFaultString();

    /**
     * Gets the locale of the fault string for this <code>SOAPFault</code>
     * object.
     * <P>
     * If this
     * <code>SOAPFault</code> is part of a message that supports SOAP 1.2 then
     * this call is equivalent to:
     * <pre>
     *    Locale locale = null;
     *    try {
     *        locale = (Locale) getFaultReasonLocales().next();
     *    } catch (SOAPException e) {}
     *    return locale;
     * </pre>
     *
     * @return a <code>Locale</code> object indicating the native language of
     *          the fault string or <code>null</code> if no locale was specified
```

```
     * @see #setFaultString(String, Locale)
     *
     * @since SAAJ 1.2
     */
    public Locale getFaultStringLocale();

    /**
     * Returns true if this <code>SOAPFault</code> has a <code>Detail</code>
     * subelement and false otherwise. Equivalent to
     * <code>(getDetail()!=null)</code>.
     *
     * @return true if this <code>SOAPFault</code> has a <code>Detail</code>
     * subelement and false otherwise.
     *
     * @since SAAJ 1.3
     */
    public boolean hasDetail();

    /**
     * Returns the optional detail element for this <code>SOAPFault</code>
     * object.
     * <P>
     * A <code>Detail</code> object carries application-specific error
     * information, the scope of the error information is restricted to
     * faults in the <code>SOAPBodyElement</code> objects if this is a
     * SOAP 1.1 Fault.
     *
     * @return a <code>Detail</code> object with application-specific
     *         error information if present, null otherwise
     */
    public Detail getDetail();

    /**
     * Creates an optional <code>Detail</code> object and sets it as the
     * <code>Detail</code> object for this <code>SOAPFault</code>
     * object.
     * <P>
     * It is illegal to add a detail when the fault already
     * contains a detail. Therefore, this method should be called
     * only after the existing detail has been removed.
     *
     * @return the new <code>Detail</code> object
     *
     * @exception SOAPException if this
     *            <code>SOAPFault</code> object already contains a
     *            valid <code>Detail</code> object
     */
    public Detail addDetail() throws SOAPException;

    /**
     * Returns an <code>Iterator</code> over a distinct sequence of
     * <code>Locale</code>s for which there are associated Reason Text items.
     * Any of these <code>Locale</code>s can be used in a call to
```

```
     * <code>getFaultReasonText</code> in order to obtain a localized version
     * of the Reason Text string.
     *
     * @return an <code>Iterator</code> over a sequence of <code>Locale</code>
     *      objects for which there are associated Reason Text items.
     *
     * @exception SOAPException if there was an error in retrieving
     * the  fault Reason locales.
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Reason.
     *
     * @since SAAJ 1.3
     */
    public Iterator getFaultReasonLocales() throws SOAPException;

    /**
     * Returns an <code>Iterator</code> over a sequence of
     * <code>String</code> objects containing all of the Reason Text items for
     * this <code>SOAPFault</code>.
     *
     * @return an <code>Iterator</code> over env:Fault/env:Reason/env:Text items.
     *
     * @exception SOAPException if there was an error in retrieving
     * the  fault Reason texts.
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Reason.
     *
     * @since SAAJ 1.3
     */
    public Iterator getFaultReasonTexts() throws SOAPException;

    /**
     * Returns the Reason Text associated with the given <code>Locale</code>.
     * If more than one such Reason Text exists the first matching Text is
     * returned
     *
     * @param locale -- the <code>Locale</code> for which a localized
     *      Reason Text is desired
     *
     * @return the Reason Text associated with <code>locale</code>
     *
     * @see #getFaultString
     *
     * @exception SOAPException if there was an error in retrieving
     * the  fault Reason text for the specified locale .
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Reason.
     *
     * @since SAAJ 1.3
     */
    public String getFaultReasonText(Locale locale) throws SOAPException;

    /**
```

```
     * Appends or replaces a Reason Text item containing the specified
     * text message and an <i>xml:lang</i> derived from
     * <code>locale</code>. If a Reason Text item with this
     * <i>xml:lang</i> already exists its text value will be replaced
     * with <code>text</code>.
     * The <code>locale</code> parameter should not be <code>null</code>
     * <P>
     * Code sample:
     *
     * <PRE>
     * SOAPFault fault = ...;
     * fault.addFaultReasonText("Version Mismatch", Locale.ENGLISH);
     * </PRE>
     *
     * @param text -- reason message string
     * @param locale -- Locale object representing the locale of the message
     *
     * @exception SOAPException if there was an error in adding the Reason text
     * or the <code>locale</code> passed was <code>null</code>.
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Reason.
     *
     * @since SAAJ 1.3
     */
    public void addFaultReasonText(String text, java.util.Locale locale)
        throws SOAPException;

    /**
     * Returns the optional Node element value for this
     * <code>SOAPFault</code> object. The Node element is
     * optional in SOAP 1.2.
     *
     * @return Content of the env:Fault/env:Node element as a String
     * or <code>null</code> if none
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Node.
     *
     * @since SAAJ 1.3
     */
    public String getFaultNode();

    /**
     * Creates or replaces any existing Node element value for
     * this <code>SOAPFault</code> object. The Node element
     * is optional in SOAP 1.2.
     *
     * @exception SOAPException  if there was an error in setting the
     *            Node for this  <code>SOAPFault</code> object.
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Node.
     *
     *
```

```
     * @since SAAJ 1.3
     */
    public void setFaultNode(String uri) throws SOAPException;

    /**
     * Returns the optional Role element value for this
     * <code>SOAPFault</code> object. The Role element is
     * optional in SOAP 1.2.
     *
     * @return Content of the env:Fault/env:Role element as a String
     * or <code>null</code> if none
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Role.
     *
     * @since SAAJ 1.3
     */
    public String getFaultRole();

    /**
     * Creates or replaces any existing Role element value for
     * this <code>SOAPFault</code> object. The Role element
     * is optional in SOAP 1.2.
     *
     * @param uri - the URI of the Role
     *
     * @exception SOAPException  if there was an error in setting the
     *            Role for this  <code>SOAPFault</code> object.
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Role.
     *
     * @since SAAJ 1.3
     */
    public void setFaultRole(String uri) throws SOAPException;

}
```

17.23 SOAPFaultElement.java

Secuencia 184: javax/xml/soap/SOAPFaultElement.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A representation of the contents in
 * a <code>SOAPFault</code> object.  The <code>Detail</code> interface
 * is a <code>SOAPFaultElement</code>.
 * <P>
 * Content is added to a <code>SOAPFaultElement</code> using the
 * <code>SOAPElement</code> method <code>addTextNode</code>.
 */
public interface SOAPFaultElement extends SOAPElement {
}
```

17.24 SOAPHeader.java

Secuencia 185: javax/xml/soap/SOAPHeader.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```

package javax.xml.soap;

import java.util.Iterator;

import javax.xml.namespace.QName;

/**
 * A representation of the SOAP header
 * element. A SOAP header element consists of XML data that affects
 * the way the application-specific content is processed by the message
 * provider. For example, transaction semantics, authentication information,
 * and so on, can be specified as the content of a <code>SOAPHeader</code>
 * object.
 * <P>
 * A <code>SOAPEnvelope</code> object contains an empty
 * <code>SOAPHeader</code> object by default. If the <code>SOAPHeader</code>
 * object, which is optional, is not needed, it can be retrieved and deleted
 * with the following line of code. The variable <i>se</i> is a
 * <code>SOAPEnvelope</code> object.
 * <PRE>
 *      se.getHeader().detachNode();
 * </PRE>
 *
 * A <code>SOAPHeader</code> object is created with the <code>SOAPEnvelope</code>
 * method <code>addHeader</code>. This method, which creates a new header and adds it
 * to the envelope, may be called only after the existing header has been removed.
 *
 * <PRE>
 *      se.getHeader().detachNode();
 *      SOAPHeader sh = se.addHeader();
 * </PRE>
 * <P>
 * A <code>SOAPHeader</code> object can have only <code>SOAPHeaderElement</code>
 * objects as its immediate children. The method <code>addHeaderElement</code>
 * creates a new <code>HeaderElement</code> object and adds it to the
 * <code>SOAPHeader</code> object. In the following line of code, the
 * argument to the method <code>addHeaderElement</code> is a <code>Name</code>
 * object that is the name for the new <code>HeaderElement</code> object.
 * <PRE>
 *      SOAPHeaderElement shElement = sh.addHeaderElement(name);
 * </PRE>
 *
 * @see SOAPHeaderElement
 */
public interface SOAPHeader extends SOAPElement {
    /**
     * Creates a new <code>SOAPHeaderElement</code> object initialized with the
     * specified name and adds it to this <code>SOAPHeader</code> object.
     *
     * @param name a <code>Name</code> object with the name of the new
     *        <code>SOAPHeaderElement</code> object
     * @return the new <code>SOAPHeaderElement</code> object that was
```

```
     *          inserted into this <code>SOAPHeader</code> object
     * @exception SOAPException if a SOAP error occurs
     * @see SOAPHeader#addHeaderElement(javax.xml.namespace.QName)
     */
    public SOAPHeaderElement addHeaderElement(Name name)
        throws SOAPException;

    /**
     * Creates a new <code>SOAPHeaderElement</code> object initialized with the
     * specified qname and adds it to this <code>SOAPHeader</code> object.
     *
     * @param qname a <code>QName</code> object with the qname of the new
     *        <code>SOAPHeaderElement</code> object
     * @return the new <code>SOAPHeaderElement</code> object that was
     *          inserted into this <code>SOAPHeader</code> object
     * @exception SOAPException if a SOAP error occurs
     * @see SOAPHeader#addHeaderElement(Name)
     * @since SAAJ 1.3
     */
    public SOAPHeaderElement addHeaderElement(QName qname)
        throws SOAPException;

    /**
     * Returns an <code>Iterator</code> over all the <code>SOAPHeaderElement</code> objects
     * in this <code>SOAPHeader</code> object
     * that have the specified <i>actor</i> and that have a MustUnderstand attribute
     * whose value is equivalent to <code>true</code>.
     * <p>
     * In SOAP 1.2 the <i>env:actor</i> attribute is replaced by the <i>env:role</i>
     * attribute, but with essentially the same semantics.
     *
     * @param actor a <code>String</code> giving the URI of the <code>actor</code> / <code>role</code>
     *        for which to search
     * @return an <code>Iterator</code> object over all the
     *         <code>SOAPHeaderElement</code> objects that contain the specified
     *          <code>actor</code> / <code>role</code> and are marked as MustUnderstand
     * @see #examineHeaderElements
     * @see #extractHeaderElements
     * @see SOAPConstants#URI_SOAP_ACTOR_NEXT
     *
     * @since SAAJ 1.2
     */
    public Iterator examineMustUnderstandHeaderElements(String actor);

    /**
     * Returns an <code>Iterator</code> over all the <code>SOAPHeaderElement</code> objects
     * in this <code>SOAPHeader</code> object
     * that have the specified <i>actor</i>.
     *
     * An <i>actor</i> is a global attribute that indicates the intermediate
     * parties that should process a message before it reaches its ultimate
     * receiver. An actor receives the message and processes it before sending
```

```
     * it on to the next actor. The default actor is the ultimate intended
     * recipient for the message, so if no actor attribute is included in a
     * <code>SOAPHeader</code> object, it is sent to the ultimate receiver
     * along with the message body.
     * <p>
     * In SOAP 1.2 the <i>env:actor</i> attribute is replaced by the <i>env:role</i>
     * attribute, but with essentially the same semantics.
     *
     * @param actor a <code>String</code> giving the URI of the <code>actor</code> / <code>role</code>
     *        for which to search
     * @return an <code>Iterator</code> object over all the
     *         <code>SOAPHeaderElement</code> objects that contain the specified
     *          <code>actor</code> / <code>role</code>
     * @see #extractHeaderElements
     * @see SOAPConstants#URI_SOAP_ACTOR_NEXT
     */
    public Iterator examineHeaderElements(String actor);

    /**
     * Returns an <code>Iterator</code> over all the <code>SOAPHeaderElement</code> objects
     * in this <code>SOAPHeader</code> object
     * that have the specified <i>actor</i> and detaches them
     * from this <code>SOAPHeader</code> object.
     * <P>
     * This method allows an actor to process the parts of the
     * <code>SOAPHeader</code> object that apply to it and to remove
     * them before passing the message on to the next actor.
     * <p>
     * In SOAP 1.2 the <i>env:actor</i> attribute is replaced by the <i>env:role</i>
     * attribute, but with essentially the same semantics.
     *
     * @param actor a <code>String</code> giving the URI of the <code>actor</code> / <code>role</code>
     *        for which to search
     * @return an <code>Iterator</code> object over all the
     *         <code>SOAPHeaderElement</code> objects that contain the specified
     *          <code>actor</code> / <code>role</code>
     *
     * @see #examineHeaderElements
     * @see SOAPConstants#URI_SOAP_ACTOR_NEXT
     */
    public Iterator extractHeaderElements(String actor);

    /**
     * Creates a new NotUnderstood <code>SOAPHeaderElement</code> object initialized
     * with the specified name and adds it to this <code>SOAPHeader</code> object.
     * This operation is supported only by SOAP 1.2.
     *
     * @param name a <code>QName</code> object with the name of the
     *        <code>SOAPHeaderElement</code> object that was not understood.
     * @return the new <code>SOAPHeaderElement</code> object that was
     *          inserted into this <code>SOAPHeader</code> object
```

```
     * @exception SOAPException if a SOAP error occurs.
     * @exception UnsupportedOperationException if this is a SOAP 1.1 Header.
     * @since SAAJ 1.3
     */
    public SOAPHeaderElement addNotUnderstoodHeaderElement(QName name)
        throws SOAPException;

    /**
     * Creates a new Upgrade <code>SOAPHeaderElement</code> object initialized
     * with the specified List of supported SOAP URIs and adds it to this
     * <code>SOAPHeader</code> object.
     * This operation is supported on both SOAP 1.1 and SOAP 1.2 header.
     *
     * @param supportedSOAPURIs an <code>Iterator</code> object with the URIs of SOAP
     *          versions supported.
     * @return the new <code>SOAPHeaderElement</code> object that was
     *          inserted into this <code>SOAPHeader</code> object
     * @exception SOAPException if a SOAP error occurs.
     * @since SAAJ 1.3
     */
    public SOAPHeaderElement addUpgradeHeaderElement(Iterator supportedSOAPURIs)
        throws SOAPException;

    /**
     * Creates a new Upgrade <code>SOAPHeaderElement</code> object initialized
     * with the specified array of supported SOAP URIs and adds it to this
     * <code>SOAPHeader</code> object.
     * This operation is supported on both SOAP 1.1 and SOAP 1.2 header.
     *
     * @param  supportedSoapUris an array of the URIs of SOAP versions supported.
     * @return the new <code>SOAPHeaderElement</code> object that was
     *          inserted into this <code>SOAPHeader</code> object
     * @exception SOAPException if a SOAP error occurs.
     * @since SAAJ 1.3
     */
    public SOAPHeaderElement addUpgradeHeaderElement(String[] supportedSoapUris)
        throws SOAPException;

    /**
     * Creates a new Upgrade <code>SOAPHeaderElement</code> object initialized
     * with the specified supported SOAP URI and adds it to this
     * <code>SOAPHeader</code> object.
     * This operation is supported on both SOAP 1.1 and SOAP 1.2 header.
     *
     * @param supportedSoapUri the URI of SOAP the version that is supported.
     * @return the new <code>SOAPHeaderElement</code> object that was
     *          inserted into this <code>SOAPHeader</code> object
     * @exception SOAPException if a SOAP error occurs.
     * @since SAAJ 1.3
     */
    public SOAPHeaderElement addUpgradeHeaderElement(String supportedSoapUri)
        throws SOAPException;

```

```
    /**
     * Returns an <code>Iterator</code> over all the <code>SOAPHeaderElement</code> objects
     * in this <code>SOAPHeader</code> object.
     *
     * @return an <code>Iterator</code> object over all the
     *          <code>SOAPHeaderElement</code> objects contained by this
     *          <code>SOAPHeader</code>
     * @see #extractAllHeaderElements
     *
     * @since SAAJ 1.2
     */
    public Iterator examineAllHeaderElements();

    /**
     * Returns an <code>Iterator</code> over all the <code>SOAPHeaderElement</code> objects
     * in this <code>SOAPHeader</code> object and detaches them
     * from this <code>SOAPHeader</code> object.
     *
     * @return an <code>Iterator</code> object over all the
     *          <code>SOAPHeaderElement</code> objects contained by this
     *          <code>SOAPHeader</code>
     *
     * @see #examineAllHeaderElements
     *
     * @since SAAJ 1.2
     */
    public Iterator extractAllHeaderElements();

}
```

17.25 SOAPHeaderElement.java

Secuencia 186: javax/xml/soap/SOAPHeaderElement.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

package javax.xml.soap;

/**
 * An object representing the contents in the SOAP header part of the
 * SOAP envelope.
 * The immediate children of a <code>SOAPHeader</code> object can
 * be represented only as <code>SOAPHeaderElement</code> objects.
 * <P>
 * A <code>SOAPHeaderElement</code> object can have other
 * <code>SOAPElement</code> objects as its children.
 */
public interface SOAPHeaderElement extends SOAPElement {

    /**
     * Sets the actor associated with this <code>SOAPHeaderElement</code>
     * object to the specified actor. The default value of an actor is:
     *          <code>SOAPConstants.URI_SOAP_ACTOR_NEXT</code>
     * <P>
     * If this <code>SOAPHeaderElement</code> supports SOAP 1.2 then this call is
     * equivalent to {@link #setRole(String)}
     *
     * @param  actorURI a <code>String</code> giving the URI of the actor
     *           to set
     *
     * @exception IllegalArgumentException if there is a problem in
     * setting the actor.
     *
     * @see #getActor
     */
    public void setActor(String actorURI);

    /**
     * Sets the <code>Role</code> associated with this <code>SOAPHeaderElement</code>
     * object to the specified <code>Role</code>.
     *
     * @param uri - the URI of the <code>Role</code>
     *
     * @throws SOAPException if there is an error in setting the role
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Role.
     *
     * @since SAAJ 1.3
     */
    public void setRole(String uri) throws SOAPException;

    /**
     * Returns the uri of the <i>actor</i> attribute of this
```

```
     * <code>SOAPHeaderElement</code>.
     *<P>
     * If this <code>SOAPHeaderElement</code> supports SOAP 1.2 then this call is
     * equivalent to {@link #getRole()}
     * @return  a <code>String</code> giving the URI of the actor
     * @see #setActor
     */
    public String getActor();

    /**
     * Returns the value of the <i>Role</i> attribute of this
     * <code>SOAPHeaderElement</code>.
     *
     * @return a <code>String</code> giving the URI of the <code>Role</code>
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Fault Role.
     *
     * @since SAAJ 1.3
     */
    public String getRole();

    /**
     * Sets the mustUnderstand attribute for this <code>SOAPHeaderElement</code>
     * object to be either true or false.
     * <P>
     * If the mustUnderstand attribute is on, the actor who receives the
     * <code>SOAPHeaderElement</code> must process it correctly. This
     * ensures, for example, that if the <code>SOAPHeaderElement</code>
     * object modifies the message, that the message is being modified correctly.
     *
     * @param mustUnderstand <code>true</code> to set the mustUnderstand
     *        attribute to true; <code>false</code> to set it to false
     *
     * @exception IllegalArgumentException if there is a problem in
     * setting the mustUnderstand attribute
     * @see #getMustUnderstand
     * @see #setRelay
     */
    public void setMustUnderstand(boolean mustUnderstand);

    /**
     * Returns the boolean value of the mustUnderstand attribute for this
     * <code>SOAPHeaderElement</code>.
     *
     * @return <code>true</code> if the mustUnderstand attribute of this
     *        <code>SOAPHeaderElement</code> object is turned on; <code>false</code>
     *         otherwise
     */
    public boolean getMustUnderstand();

    /**
     * Sets the <i>relay</i> attribute for this <code>SOAPHeaderElement</code> to be
```

```
     * either true or false.
     * <P>
     * The SOAP relay attribute is set to true to indicate that the SOAP header
     * block must be relayed by any node that is targeted by the header block
     * but not actually process it. This attribute is ignored on header blocks
     * whose mustUnderstand attribute is set to true or that are targeted at
     * the ultimate reciever (which is the default). The default value of this
     * attribute is <code>false</code>.
     *
     * @param relay the new value of the <i>relay</i> attribute
     *
     * @exception SOAPException if there is a problem in setting the
     * relay attribute.
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Relay attribute.
     *
     * @see #setMustUnderstand
     * @see #getRelay
     *
     * @since SAAJ 1.3
     */
    public void setRelay(boolean relay) throws SOAPException;

    /**
     * Returns the boolean value of the <i>relay</i> attribute for this
     * <code>SOAPHeaderElement</code>
     *
     * @return <code>true</code> if the relay attribute is turned on;
     * <code>false</code> otherwise
     *
     * @exception UnsupportedOperationException if this message does not
     *      support the SOAP 1.2 concept of Relay attribute.
     *
     * @see #getMustUnderstand
     * @see #setRelay
     *
     * @since SAAJ 1.3
     */
    public boolean getRelay();
}
```

17.26 SOAPMessage.java

Secuencia 187: javax/xml/soap/SOAPMessage.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;
import java.io.OutputStream;
import java.io.IOException;

import java.util.Iterator;

import javax.activation.DataHandler;

/**
 * The root class for all SOAP messages. As transmitted on the "wire", a SOAP
 * message is an XML document or a MIME message whose first body part is an
 * XML/SOAP document.
 * <P>
 * A <code>SOAPMessage</code> object consists of a SOAP part and optionally
 * one or more attachment parts. The SOAP part for a <code>SOAPMessage</code>
 * object is a <code>SOAPPart</code> object, which contains information used
 * for message routing and identification, and which can contain
 * application-specific content. All data in the SOAP Part of a message must be
 * in XML format.
 * <P>
 * A new <code>SOAPMessage</code> object contains the following by default:
 * <UL>
 *   <LI>A <code>SOAPPart</code> object
 *   <LI>A <code>SOAPEnvelope</code> object
 *   <LI>A <code>SOAPBody</code> object
 *   <LI>A <code>SOAPHeader</code> object
 * </UL>
 * The SOAP part of a message can be retrieved by calling the method <code>SOAPMessage.getSOAPPart
     ()</code>.
 * The <code>SOAPEnvelope</code> object is retrieved from the <code>SOAPPart</code>
 * object, and the <code>SOAPEnvelope</code> object is used to retrieve the
 * <code>SOAPBody</code> and <code>SOAPHeader</code> objects.
 *
 * <PRE>
 *     SOAPPart sp = message.getSOAPPart();
 *     SOAPEnvelope se = sp.getEnvelope();
 *     SOAPBody sb = se.getBody();
```

```
 *     SOAPHeader sh = se.getHeader();
 * </PRE>
 *
 * <P>
 * In addition to the mandatory <code>SOAPPart</code> object, a <code>SOAPMessage</code>
 * object may contain zero or more <code>AttachmentPart</code> objects, each
 * of which contains application-specific data. The <code>SOAPMessage</code>
 * interface provides methods for creating <code>AttachmentPart</code>
 * objects and also for adding them to a <code>SOAPMessage</code> object. A
 * party that has received a <code>SOAPMessage</code> object can examine its
 * contents by retrieving individual attachment parts.
 * <P>
 * Unlike the rest of a SOAP message, an attachment is not required to be in
 * XML format and can therefore be anything from simple text to an image file.
 * Consequently, any message content that is not in XML format must be in an
 * <code>AttachmentPart</code> object.
 * <P>
 * A <code>MessageFactory</code> object may create <code>SOAPMessage</code>
 * objects with behavior that is specialized to a particular implementation or
 * application of SAAJ. For instance, a <code>MessageFactory</code> object
 * may produce <code>SOAPMessage</code> objects that conform to a particular
 * Profile such as ebXML. In this case a <code>MessageFactory</code> object
 * might produce <code>SOAPMessage</code> objects that are initialized with
 * ebXML headers.
 * <P>
 * In order to ensure backward source compatibility, methods that are added to
 * this class after version 1.1 of the SAAJ specification are all concrete
 * instead of abstract and they all have default implementations. Unless
 * otherwise noted in the JavaDocs for those methods the default
 * implementations simply throw an <code>UnsupportedOperationException</code>
 * and the SAAJ implementation code must override them with methods that
 * provide the specified behavior. Legacy client code does not have this
 * restriction, however, so long as there is no claim made that it conforms to
 * some later version of the specification than it was originally written for.
 * A legacy class that extends the SOAPMessage class can be compiled and/or run
 * against succeeding versions of the SAAJ API without modification. If such a
 * class was correctly implemented then it will continue to behave correctly
 * relative to the version of the specification against which it was written.
 *
 * @see MessageFactory
 * @see AttachmentPart
 */
public abstract class SOAPMessage {
    /**
         * Specifies the character type encoding for the SOAP Message. Valid values
         * include "utf-8" and "utf-16". See vendor documentation for additional
         * supported values. The default is "utf-8".
         *
         * @see SOAPMessage#setProperty(String, Object) SOAPMessage.setProperty
         * @since SAAJ 1.2
         */
    public static final String CHARACTER_SET_ENCODING =
        "javax.xml.soap.character-set-encoding";
```

```java

    /**
     * Specifies whether the SOAP Message will contain an XML declaration when
     * it is sent. The only valid values are "true" and "false". The default is
     * "false".
     *
     * @see SOAPMessage#setProperty(String, Object) SOAPMessage.setProperty
     * @since SAAJ 1.2
     */
    public static final String WRITE_XML_DECLARATION =
        "javax.xml.soap.write-xml-declaration";

    /**
     * Sets the description of this <code>SOAPMessage</code> object's
     * content with the given description.
     *
     * @param description a <code>String</code> describing the content of this
     *         message
     * @see #getContentDescription
     */
    public abstract void setContentDescription(String description);

    /**
     * Retrieves a description of this <code>SOAPMessage</code> object's
     * content.
     *
     * @return a <code>String</code> describing the content of this
     *         message or <code>null</code> if no description has been set
     * @see #setContentDescription
     */
    public abstract String getContentDescription();

    /**
         * Gets the SOAP part of this <code>SOAPMessage</code> object.
         * <P>
         * <code>SOAPMessage</code> object contains one or more attachments, the
         * SOAP Part must be the first MIME body part in the message.
         *
         * @return the <code>SOAPPart</code> object for this <code>SOAPMessage</code>
         *         object
         */
    public abstract SOAPPart getSOAPPart();

    /**
         * Gets the SOAP Body contained in this <code>SOAPMessage</code> object.
         * <p>
         *
         * @return the <code>SOAPBody</code> object contained by this <code>SOAPMessage</code>
         *         object
         * @exception SOAPException
         *               if the SOAP Body does not exist or cannot be retrieved
         * @since SAAJ 1.2
         */
```

```java
    public SOAPBody getSOAPBody() throws SOAPException {
        throw new UnsupportedOperationException("getSOAPBody must be overridden by all subclasses of SOAPMessage");
    }

    /**
     * Gets the SOAP Header contained in this <code>SOAPMessage</code>
     * object.
     * <p>
     *
     * @return the <code>SOAPHeader</code> object contained by this <code>SOAPMessage</code>
     *         object
     * @exception SOAPException
     *               if the SOAP Header does not exist or cannot be retrieved
     * @since SAAJ 1.2
     */
    public SOAPHeader getSOAPHeader() throws SOAPException {
        throw new UnsupportedOperationException("getSOAPHeader must be overridden by all subclasses of SOAPMessage");
    }

    /**
     * Removes all <code>AttachmentPart</code> objects that have been added
     * to this <code>SOAPMessage</code> object.
     * <P>
     * This method does not touch the SOAP part.
     */
    public abstract void removeAllAttachments();

    /**
     * Gets a count of the number of attachments in this message. This count
     * does not include the SOAP part.
     *
     * @return the number of <code>AttachmentPart</code> objects that are
     *         part of this <code>SOAPMessage</code> object
     */
    public abstract int countAttachments();

    /**
     * Retrieves all the <code>AttachmentPart</code> objects that are part of
     * this <code>SOAPMessage</code> object.
     *
     * @return an iterator over all the attachments in this message
     */
    public abstract Iterator getAttachments();

    /**
     * Retrieves all the <code>AttachmentPart</code> objects that have header
     * entries that match the specified headers. Note that a returned
     * attachment could have headers in addition to those specified.
     *
     * @param headers
     *           a <code>MimeHeaders</code> object containing the MIME
```

```
         *           headers for which to search
         * @return an iterator over all attachments that have a header that matches
         *         one of the given headers
         */
    public abstract Iterator getAttachments(MimeHeaders headers);

    /**
     * Removes all the <code>AttachmentPart</code> objects that have header
     * entries that match the specified headers. Note that the removed
     * attachment could have headers in addition to those specified.
     *
     * @param headers
     *           a <code>MimeHeaders</code> object containing the MIME
     *           headers for which to search
     * @since SAAJ 1.3
     */
    public abstract void removeAttachments(MimeHeaders headers);


    /**
     * Returns an <code>AttachmentPart</code> object that is associated with an
     * attachment that is referenced by this <code>SOAPElement</code> or
     * <code>null</code> if no such attachment exists. References can be made
     * via an <code>href</code> attribute as described in
     * {@link <a href="http://www.w3.org/TR/SOAP-attachments#SOAPReferenceToAttachements">SOAP
       Messages with Attachments</a>},
     * or via a single <code>Text</code> child node containing a URI as
     * described in the WS-I Attachments Profile 1.0 for elements of schema
     * type {@link <a href="http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html">
       ref:swaRef</a>}.  These two mechanisms must be supported.
     * The support for references via <code>href</code> attribute also implies that
     * this method should also be supported on an element that is an
     * <i>xop:Include</i> element (
     * {@link <a  href="http://www.w3.org/2000/xp/Group/3/06/Attachments/XOP.html">XOP</a>}).
     * other reference mechanisms may be supported by individual
     * implementations of this standard. Contact your vendor for details.
     *
     * @param  element The <code>SOAPElement</code> containing the reference to an Attachment
     * @return the referenced <code>AttachmentPart</code> or null if no such
     *          <code>AttachmentPart</code> exists or no reference can be
     *          found in this <code>SOAPElement</code>.
     * @throws SOAPException if there is an error in the attempt to access the
     *          attachment
     *
     * @since SAAJ 1.3
     */
    public abstract AttachmentPart getAttachment(SOAPElement element) throws SOAPException;


    /**
     * Adds the given <code>AttachmentPart</code> object to this <code>SOAPMessage</code>
     * object. An <code>AttachmentPart</code> object must be created before
     * it can be added to a message.
```

```
     *
     * @param AttachmentPart
     *           an <code>AttachmentPart</code> object that is to become part
     *           of this <code>SOAPMessage</code> object
     * @exception IllegalArgumentException
     */
    public abstract void addAttachmentPart(AttachmentPart AttachmentPart);

    /**
     * Creates a new empty <code>AttachmentPart</code> object. Note that the
     * method <code>addAttachmentPart</code> must be called with this new
     * <code>AttachmentPart</code> object as the parameter in order for it to
     * become an attachment to this <code>SOAPMessage</code> object.
     *
     * @return a new <code>AttachmentPart</code> object that can be populated
     *         and added to this <code>SOAPMessage</code> object
     */
    public abstract AttachmentPart createAttachmentPart();

    /**
     * Creates an <code>AttachmentPart</code> object and populates it using
     * the given <code>DataHandler</code> object.
     *
     * @param dataHandler
     *           the <code>javax.activation.DataHandler</code> object that
     *           will generate the content for this <code>SOAPMessage</code>
     *           object
     * @return a new <code>AttachmentPart</code> object that contains data
     *         generated by the given <code>DataHandler</code> object
     * @exception IllegalArgumentException
     *               if there was a problem with the specified <code>DataHandler</code>
     *               object
     * @see javax.activation.DataHandler
     * @see javax.activation.DataContentHandler
     */
    public AttachmentPart createAttachmentPart(DataHandler dataHandler) {
        AttachmentPart attachment = createAttachmentPart();
        attachment.setDataHandler(dataHandler);
        return attachment;
    }

    /**
     * Returns all the transport-specific MIME headers for this <code>SOAPMessage</code>
     * object in a transport-independent fashion.
     *
     * @return a <code>MimeHeaders</code> object containing the <code>MimeHeader</code>
     *         objects
     */
    public abstract MimeHeaders getMimeHeaders();

    /**
     * Creates an <code>AttachmentPart</code> object and populates it with
     * the specified data of the specified content type. The type of the
```

```
     * <code>Object</code> should correspond to the value given for the
     * <code>Content-Type</code>.
     *
     * @param content
     *           an <code>Object</code> containing the content for the
     *           <code>AttachmentPart</code> object to be created
     * @param contentType
     *           a <code>String</code> object giving the type of content;
     *           examples are "text/xml", "text/plain", and "image/jpeg"
     * @return a new <code>AttachmentPart</code> object that contains the
     *         given data
     * @exception IllegalArgumentException
     *               may be thrown if the contentType does not match the type
     *               of the content object, or if there was no
     *               <code>DataContentHandler</code> object for the given
     *               content object
     * @see javax.activation.DataHandler
     * @see javax.activation.DataContentHandler
     */
    public AttachmentPart createAttachmentPart(
        Object content,
        String contentType) {
        AttachmentPart attachment = createAttachmentPart();
        attachment.setContent(content, contentType);
        return attachment;
    }

    /**
     * Updates this <code>SOAPMessage</code> object with all the changes that
     * have been made to it. This method is called automatically when
     * {@link SOAPMessage#writeTo(OutputStream)} is  called. However, if
     * changes are made to a message that was received or to one that has
     * already been sent, the method <code>saveChanges</code> needs to be
     * called explicitly in order to save the changes. The method <code>saveChanges</code>
     * also generates any changes that can be read back (for example, a
     * MessageId in profiles that support a message id). All MIME headers in a
     * message that is created for sending purposes are guaranteed to have
     * valid values only after <code>saveChanges</code> has been called.
     * <P>
     * In addition, this method marks the point at which the data from all
     * constituent <code>AttachmentPart</code> objects are pulled into the
     * message.
     * <P>
     *
     * @exception <code>SOAPException</code> if there was a problem saving
     *               changes to this message.
     */
    public abstract void saveChanges() throws SOAPException;

    /**
     * Indicates whether this <code>SOAPMessage</code> object needs to have
     * the method <code>saveChanges</code> called on it.
     *
```

```
     * @return <code>true</code> if <code>saveChanges</code> needs to be
     *         called; <code>false</code> otherwise.
     */
    public abstract boolean saveRequired();

    /**
     * Writes this <code>SOAPMessage</code> object to the given output
     * stream. The externalization format is as defined by the SOAP 1.1 with
     * Attachments specification.
     * <P>
     * If there are no attachments, just an XML stream is written out. For
     * those messages that have attachments, <code>writeTo</code> writes a
     * MIME-encoded byte stream.
     * <P>
     * Note that this method does not write the transport-specific MIME Headers
     * of the Message
     *
     * @param out
     *           the <code>OutputStream</code> object to which this <code>SOAPMessage</code>
     *           object will be written
     * @exception IOException
     *               if an I/O error occurs
     * @exception SOAPException
     *               if there was a problem in externalizing this SOAP message
     */
    public abstract void writeTo(OutputStream out)
        throws SOAPException, IOException;

    /**
     * Associates the specified value with the specified property. If there was
     * already a value associated with this property, the old value is
     * replaced.
     * <p>
     * The valid property names include
     * {@link SOAPMessage#WRITE_XML_DECLARATION}  and
     * {@link SOAPMessage#CHARACTER_SET_ENCODING}. All of these standard SAAJ
     * properties are prefixed by "javax.xml.soap". Vendors may also add
     * implementation specific properties. These properties must be prefixed
     * with package names that are unique to the vendor.
     * <p>
     * Setting the property <code>WRITE_XML_DECLARATION</code> to <code>"true"</code>
     * will cause an XML Declaration to be written out at the start of the SOAP
     * message. The default value of "false" suppresses this declaration.
     * <p>
     * The property <code>CHARACTER_SET_ENCODING</code> defaults to the value
     * <code>"utf-8"</code> which causes the SOAP message to be encoded using
     * UTF-8. Setting <code>CHARACTER_SET_ENCODING</code> to <code>"utf-16"</code>
     * causes the SOAP message to be encoded using UTF-16.
     * <p>
     * Some implementations may allow encodings in addition to UTF-8 and
     * UTF-16. Refer to your vendor's documentation for details.
     *
     * @param property
```

```
     *           the property with which the specified value is to be
     *           associated.
     * @param value
     *           the value to be associated with the specified property
     * @exception SOAPException
     *               if the property name is not recognized.
     * @since SAAJ 1.2
     */
    public void setProperty(String property, Object value)
        throws SOAPException {
            throw new UnsupportedOperationException("setProperty must be overridden by all
                subclasses of SOAPMessage");
    }

    /**
     * Retrieves value of the specified property.
     *
     * @param property
     *           the name of the property to retrieve
     * @return the value associated with the named property or <code>null</code>
     *         if no such property exists.
     * @exception SOAPException
     *               if the property name is not recognized.
     * @since SAAJ 1.2
     */
    public Object getProperty(String property) throws SOAPException {
        throw new UnsupportedOperationException("getProperty must be overridden by all subclasses
            of SOAPMessage");
    }
}
```

17.27 SOAPPart.java

Secuencia 188: javax/xml/soap/SOAPPart.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

package javax.xml.soap;

import java.util.Iterator;

import javax.xml.transform.Source;

/**
 * The container for the SOAP-specific portion of a <code>SOAPMessage</code>
 * object. All messages are required to have a SOAP part, so when a
 * <code>SOAPMessage</code> object is created, it will automatically
 * have a <code>SOAPPart</code> object.
 *<P>
 * A <code>SOAPPart</code> object is a MIME part and has the MIME headers
 * Content-Id, Content-Location, and Content-Type.  Because the value of
 * Content-Type must be "text/xml", a <code>SOAPPart</code> object automatically
 * has a MIME header of Content-Type with its value set to "text/xml".
 * The value must be "text/xml" because content in the SOAP part of a
 * message must be in XML format.  Content that is not of type "text/xml"
 * must be in an <code>AttachmentPart</code> object rather than in the
 * <code>SOAPPart</code> object.
 * <P>
 * When a message is sent, its SOAP part must have the MIME header Content-Type
 * set to "text/xml". Or, from the other perspective, the SOAP part of any
 * message that is received must have the MIME header Content-Type with a
 * value of "text/xml".
 * <P>
 * A client can access the <code>SOAPPart</code> object of a
 * <code>SOAPMessage</code> object by
 * calling the method <code>SOAPMessage.getSOAPPart</code>. The
 * following  line of code, in which <code>message</code> is a
 * <code>SOAPMessage</code> object, retrieves the SOAP part of a message.
 * <PRE>
 *   SOAPPart soapPart = message.getSOAPPart();
 * </PRE>
 * <P>
 * A <code>SOAPPart</code> object contains a <code>SOAPEnvelope</code> object,
 * which in turn contains a <code>SOAPBody</code> object and a
 * <code>SOAPHeader</code> object.
 * The <code>SOAPPart</code> method <code>getEnvelope</code> can be used
 * to retrieve the <code>SOAPEnvelope</code> object.
 * <P>
 */
public abstract class SOAPPart implements org.w3c.dom.Document, Node {

    /**
     * Gets the <code>SOAPEnvelope</code> object associated with this
     * <code>SOAPPart</code> object. Once the SOAP envelope is obtained, it
```

```
     * can be used to get its contents.
     *
     * @return the <code>SOAPEnvelope</code> object for this
     *           <code>SOAPPart</code> object
     * @exception SOAPException if there is a SOAP error
     */
    public abstract SOAPEnvelope getEnvelope() throws SOAPException;

    /**
     * Retrieves the value of the MIME header whose name is "Content-Id".
     *
     * @return a <code>String</code> giving the value of the MIME header
     *         named "Content-Id"
     * @see #setContentId
     */
    public String getContentId() {
        String[] values = getMimeHeader("Content-Id");
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    /**
     * Retrieves the value of the MIME header whose name is "Content-Location".
     *
     * @return a <code>String</code> giving the value of the MIME header whose
     *          name is "Content-Location"
     * @see #setContentLocation
     */
    public String getContentLocation() {
        String[] values = getMimeHeader("Content-Location");
        if (values != null && values.length > 0)
            return values[0];
        return null;
    }

    /**
     * Sets the value of the MIME header named "Content-Id"
     * to the given <code>String</code>.
     *
     * @param contentId a <code>String</code> giving the value of the MIME
     *        header "Content-Id"
     *
     * @exception IllegalArgumentException if there is a problem in
     * setting the content id
     * @see #getContentId
     */
    public void setContentId(String contentId)
    {
        setMimeHeader("Content-Id", contentId);
    }
    /**
     * Sets the value of the MIME header "Content-Location"
```

```
     * to the given <code>String</code>.
     *
     * @param contentLocation a <code>String</code> giving the value
     *        of the MIME
     *        header "Content-Location"
     * @exception IllegalArgumentException if there is a problem in
     *            setting the content location.
     * @see #getContentLocation
     */
    public void setContentLocation(String contentLocation)
    {
        setMimeHeader("Content-Location", contentLocation);
    }
    /**
     * Removes all MIME headers that match the given name.
     *
     * @param header a <code>String</code> giving the name of the MIME header(s) to
     *               be removed
     */
    public abstract void removeMimeHeader(String header);

    /**
     * Removes all the <code>MimeHeader</code> objects for this
     * <code>SOAPEnvelope</code> object.
     */
    public abstract void removeAllMimeHeaders();

    /**
     * Gets all the values of the <code>MimeHeader</code> object
     * in this <code>SOAPPart</code> object that
     * is identified by the given <code>String</code>.
     *
     * @param name the name of the header; example: "Content-Type"
     * @return a <code>String</code> array giving all the values for the
     *         specified header
     * @see #setMimeHeader
     */
    public abstract String[] getMimeHeader(String name);

    /**
     * Changes the first header entry that matches the given header name
     * so that its value is the given value, adding a new header with the
     * given name and value if no
     * existing header is a match. If there is a match, this method clears
     * all existing values for the first header that matches and sets the
     * given value instead. If more than one header has
     * the given name, this method removes all of the matching headers after
     * the first one.
     * <P>
     * Note that RFC822 headers can contain only US-ASCII characters.
     *
     * @param   name    a <code>String</code> giving the header name
     *                  for which to search
```

```
     * @param   value   a <code>String</code> giving the value to be set.
     *                  This value will be substituted for the current value(s)
     *                  of the first header that is a match if there is one.
     *                  If there is no match, this value will be the value for
     *                  a new <code>MimeHeader</code> object.
     *
     * @exception IllegalArgumentException if there was a problem with
     *            the specified mime header name or value
     * @see #getMimeHeader
     */
    public abstract void setMimeHeader(String name, String value);

    /**
     * Creates a <code>MimeHeader</code> object with the specified
     * name and value and adds it to this <code>SOAPPart</code> object.
     * If a <code>MimeHeader</code> with the specified name already
     * exists, this method adds the specified value to the already
     * existing value(s).
     * <P>
     * Note that RFC822 headers can contain only US-ASCII characters.
     *
     * @param   name    a <code>String</code> giving the header name
     * @param   value   a <code>String</code> giving the value to be set
     *                  or added
     * @exception IllegalArgumentException if there was a problem with
     *            the specified mime header name or value
     */
    public abstract void addMimeHeader(String name, String value);

    /**
     * Retrieves all the headers for this <code>SOAPPart</code> object
     * as an iterator over the <code>MimeHeader</code> objects.
     *
     * @return  an <code>Iterator</code> object with all of the Mime
     *          headers for this <code>SOAPPart</code> object
     */
    public abstract Iterator getAllMimeHeaders();

    /**
     * Retrieves all <code>MimeHeader</code> objects that match a name in
     * the given array.
     *
     * @param names a <code>String</code> array with the name(s) of the
     *        MIME headers to be returned
     * @return  all of the MIME headers that match one of the names in the
     *           given array, returned as an <code>Iterator</code> object
     */
    public abstract Iterator getMatchingMimeHeaders(String[] names);

    /**
     * Retrieves all <code>MimeHeader</code> objects whose name does
     * not match a name in the given array.
     *
```

```
     * @param names a <code>String</code> array with the name(s) of the
     *        MIME headers not to be returned
     * @return  all of the MIME headers in this <code>SOAPPart</code> object
     *          except those that match one of the names in the
     *           given array.  The nonmatching MIME headers are returned as an
     *           <code>Iterator</code> object.
     */
    public abstract Iterator getNonMatchingMimeHeaders(String[] names);

    /**
     * Sets the content of the <code>SOAPEnvelope</code> object with the data
     * from the given <code>Source</code> object. This <code>Source</code>
     * must contain a valid SOAP document.
     *
     * @param source the <code>javax.xml.transform.Source</code> object with the
     *        data to be set
     *
     * @exception SOAPException if there is a problem in setting the source
     * @see #getContent
     */
    public abstract void setContent(Source source) throws SOAPException;

    /**
     * Returns the content of the SOAPEnvelope as a JAXP <code>Source</code>
     * object.
     *
     * @return the content as a <code>javax.xml.transform.Source</code> object
     *
     * @exception SOAPException if the implementation cannot convert
     *                          the specified <code>Source</code> object
     * @see #setContent
     */
    public abstract Source getContent() throws SOAPException;
}
```

17.28 Text.java

Secuencia 189: javax/xml/soap/Text.java

```
/*
 * Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.soap;

/**
 * A representation of a node whose value is text.  A <code>Text</code> object
 * may represent text that is content or text that is a comment.
 *
 */
public interface Text extends Node, org.w3c.dom.Text {

    /**
     * Retrieves whether this <code>Text</code> object represents a comment.
     *
     * @return <code>true</code> if this <code>Text</code> object is a
     *         comment; <code>false</code> otherwise
     */
    public boolean isComment();
}
```

18 Xml Stream (javax.xml.stream package)

18.1 EventFilter.java

Secuencia 190: javax/xml/stream/EventFilter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.stream.events.XMLEvent;

/**
 * This interface declares a simple filter interface that one can
 * create to filter XMLEventReaders
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface EventFilter {
  /**
   * Tests whether this event is part of this stream.  This method
   * will return true if this filter accepts this event and false
   * otherwise.
   *
   * @param event the event to test
   * @return true if this filter accepts this event, false otherwise
   */
  public boolean accept(XMLEvent event);
}
```

18.2 FactoryConfigurationError.java

Secuencia 191: javax/xml/stream/FactoryConfigurationError.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * An error class for reporting factory configuration errors.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public class FactoryConfigurationError extends Error {
    private static final long serialVersionUID = -2994412584589975744L;

  Exception nested;

  /**
   * Default constructor
   */
  public FactoryConfigurationError(){}

  /**
   * Construct an exception with a nested inner exception
   *
   * @param e the exception to nest
   */
  public FactoryConfigurationError(java.lang.Exception e){
    nested = e;
  }

  /**
   * Construct an exception with a nested inner exception
   * and a message
   *
   * @param e the exception to nest
   * @param msg the message to report
   */
  public FactoryConfigurationError(java.lang.Exception e, java.lang.String msg){
    super(msg);
    nested = e;
  }

  /**
   * Construct an exception with a nested inner exception
   * and a message
   *
```

```
   * @param msg the message to report
   * @param e the exception to nest
   */
  public FactoryConfigurationError(java.lang.String msg, java.lang.Exception e){
    super(msg);
    nested = e;
  }

  /**
   * Construct an exception with associated message
   *
   * @param msg the message to report
   */
  public FactoryConfigurationError(java.lang.String msg) {
    super(msg);
  }

  /**
   * Return the nested exception (if any)
   *
   * @return the nested exception or null
   */
  public Exception getException() {
    return nested;
  }
    /**
     * use the exception chaining mechanism of JDK1.4
    */
    @Override
    public Throwable getCause() {
        return nested;
    }

  /**
   * Report the message associated with this error
   *
   * @return the string value of the message
   */
  public String getMessage() {
    String msg = super.getMessage();
    if(msg != null)
      return msg;
    if(nested != null){
      msg = nested.getMessage();
      if(msg == null)
        msg = nested.getClass().toString();
    }
    return msg;
  }



}
```

18.3 FactoryFinder.java

Secuencia 192: javax/xml/stream/FactoryFinder.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.stream;

import java.io.File;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Iterator;
import java.util.Properties;
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * <p>Implements pluggable streams.</p>
 *
 * <p>This class is duplicated for each JAXP subpackage so keep it in
 * sync.  It is package private for secure class loading.</p>
 *
 * @author Santiago.PericasGeertsen@sun.com
 */
class FactoryFinder {
    // Check we have access to package.
    private static final String DEFAULT_PACKAGE = "com.sun.xml.internal.";

    /**
     * Internal debug flag.
     */
```

```java
    private static boolean debug = false;

    /**
     * Cache for properties in java.home/lib/jaxp.properties
     */
    final private static Properties cacheProps = new Properties();

    /**
     * Flag indicating if properties from java.home/lib/jaxp.properties
     * have been cached.
     */
    private static volatile boolean firstTime = true;

    /**
     * Security support class use to check access control before
     * getting certain system resources.
     */
    final private static SecuritySupport ss = new SecuritySupport();

    // Define system property "jaxp.debug" to get output
    static {
        // Use try/catch block to support applets, which throws
        // SecurityException out of this code.
        try {
            String val = ss.getSystemProperty("jaxp.debug");
            // Allow simply setting the prop to turn on debug
            debug = val != null && !"false".equals(val);
        }
        catch (SecurityException se) {
            debug = false;
        }
    }

    private static void dPrint(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * Attempt to load a class using the class loader supplied. If that fails
     * and fall back is enabled, the current (i.e. bootstrap) class loader is
     * tried.
     *
     * If the class loader supplied is <code>null</code>, first try using the
     * context class loader followed by the current (i.e. bootstrap) class
     * loader.
     *
     * Use bootstrap classLoader if cl = null and useBSClsLoader is true
     */
    static private Class getProviderClass(String className, ClassLoader cl,
            boolean doFallback, boolean useBSClsLoader) throws ClassNotFoundException
    {
```

```
        try {
            if (cl == null) {
                if (useBSClsLoader) {
                    return Class.forName(className, false, FactoryFinder.class.getClassLoader());
                } else {
                    cl = ss.getContextClassLoader();
                    if (cl == null) {
                        throw new ClassNotFoundException();
                    }
                    else {
                        return Class.forName(className, false, cl);
                    }
                }
            }
            else {
                return Class.forName(className, false, cl);
            }
        }
        catch (ClassNotFoundException e1) {
            if (doFallback) {
                // Use current class loader - should always be bootstrap CL
                return Class.forName(className, false, FactoryFinder.class.getClassLoader());
            }
            else {
                throw e1;
            }
        }
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl, boolean doFallback)
        throws FactoryConfigurationError
    {
        return newInstance(type, className, cl, doFallback, false);
    }

    /**
     * Create an instance of a class. Delegates to method
```

```
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     *
     * @param useBSClsLoader True if cl=null actually meant bootstrap classLoader. This parameter
     * is needed since DocumentBuilderFactory/SAXParserFactory defined null as context classLoader.
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl,
                             boolean doFallback, boolean useBSClsLoader)
        throws FactoryConfigurationError
    {
        assert type != null;

        // make sure we have access to restricted packages
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                cl = null;
                useBSClsLoader = true;
            }
        }

        try {
            Class<?> providerClass = getProviderClass(className, cl, doFallback, useBSClsLoader);
            if (!type.isAssignableFrom(providerClass)) {
                throw new ClassCastException(className + " cannot be cast to " + type.getName());
            }
            Object instance = providerClass.newInstance();
            if (debug) {    // Extra check to avoid computing cl strings
                dPrint("created new instance of " + providerClass +
                       " using ClassLoader: " + cl);
            }
            return type.cast(instance);
        }
        catch (ClassNotFoundException x) {
            throw new FactoryConfigurationError(
                "Provider " + className + " not found", x);
        }
        catch (Exception x) {
            throw new FactoryConfigurationError(
                "Provider " + className + " could not be instantiated: " + x,
                x);
        }
    }
```

```java

    /**
     * Finds the implementation Class object in the specified order.
     *
     * @return Class object of factory, never null
     *
     * @param type                  Base class / Service interface  of the
     *                              factory to find.
     *
     * @param fallbackClassName     Implementation class name, if nothing else
     *                              is found.  Use null to mean no fallback.
     *
     * Package private so this code can be shared.
     */
    static <T> T find(Class<T> type, String fallbackClassName)
        throws FactoryConfigurationError
    {
        return find(type, type.getName(), null, fallbackClassName);
    }

    /**
     * Finds the implementation Class object in the specified order.  Main
     * entry point.
     * @return Class object of factory, never null
     *
     * @param type                  Base class / Service interface  of the
     *                              factory to find.
     *
     * @param factoryId             Name of the factory to find, same as
     *                              a property name
     *
     * @param cl                    ClassLoader to be used to load the class, null means to use
     * the bootstrap ClassLoader
     *
     * @param fallbackClassName     Implementation class name, if nothing else
     *                              is found.  Use null to mean no fallback.
     *
     * Package private so this code can be shared.
     */
    static <T> T find(Class<T> type, String factoryId, ClassLoader cl, String fallbackClassName)
        throws FactoryConfigurationError
    {
        dPrint("find factoryId =" + factoryId);

        // Use the system property first
        try {

            final String systemProp;
            if (type.getName().equals(factoryId)) {
                systemProp = ss.getSystemProperty(factoryId);
            } else {
                systemProp = System.getProperty(factoryId);
            }
```

```
            if (systemProp != null) {
                dPrint("found system property, value=" + systemProp);
                return newInstance(type, systemProp, cl, true);
            }
        }
        catch (SecurityException se) {
            throw new FactoryConfigurationError(
                    "Failed to read factoryId '" + factoryId + "'", se);
        }

        // Try read $java.home/lib/stax.properties followed by
        // $java.home/lib/jaxp.properties if former not present
        String configFile = null;
        try {
            if (firstTime) {
                synchronized (cacheProps) {
                    if (firstTime) {
                        configFile = ss.getSystemProperty("java.home") + File.separator +
                            "lib" + File.separator + "stax.properties";
                        File f = new File(configFile);
                        firstTime = false;
                        if (ss.doesFileExist(f)) {
                            dPrint("Read properties file "+f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                        else {
                            configFile = ss.getSystemProperty("java.home") + File.separator +
                                "lib" + File.separator + "jaxp.properties";
                            f = new File(configFile);
                            if (ss.doesFileExist(f)) {
                                dPrint("Read properties file "+f);
                                cacheProps.load(ss.getFileInputStream(f));
                            }
                        }
                    }
                }
            }
            final String factoryClassName = cacheProps.getProperty(factoryId);

            if (factoryClassName != null) {
                dPrint("found in " + configFile + " value=" + factoryClassName);
                return newInstance(type, factoryClassName, cl, true);
            }
        }
        catch (Exception ex) {
            if (debug) ex.printStackTrace();
        }

        if (type.getName().equals(factoryId)) {
            // Try Jar Service Provider Mechanism
            final T provider = findServiceProvider(type, cl);
            if (provider != null) {
                return provider;
```

```
            }
        } else {
            // We're in the case where a 'custom' factoryId was provided,
            // and in every case where that happens, we expect that
            // fallbackClassName will be null.
            assert fallbackClassName == null;
        }
        if (fallbackClassName == null) {
            throw new FactoryConfigurationError(
                "Provider for " + factoryId + " cannot be found", null);
        }

        dPrint("loaded from fallback value: " + fallbackClassName);
        return newInstance(type, fallbackClassName, cl, true);
    }

    /*
     * Try to find provider using the ServiceLoader API
     *
     * @param type Base class / Service interface  of the factory to find.
     *
     * @return instance of provider class if found or null
     */
    private static <T> T findServiceProvider(final Class<T> type, final ClassLoader cl) {
        try {
            return AccessController.doPrivileged(new PrivilegedAction<T>() {
                @Override
                public T run() {
                    final ServiceLoader<T> serviceLoader;
                    if (cl == null) {
                        //the current thread's context class loader
                        serviceLoader = ServiceLoader.load(type);
                    } else {
                        serviceLoader = ServiceLoader.load(type, cl);
                    }
                    final Iterator<T> iterator = serviceLoader.iterator();
                    if (iterator.hasNext()) {
                        return iterator.next();
                    } else {
                        return null;
                    }
                }
            });
        } catch(ServiceConfigurationError e) {
            // It is not possible to wrap an error directly in
            // FactoryConfigurationError - so we need to wrap the
            // ServiceConfigurationError in a RuntimeException.
            // The alternative would be to modify the logic in
            // FactoryConfigurationError to allow setting a
            // Throwable as the cause, but that could cause
            // compatibility issues down the road.
            final RuntimeException x = new RuntimeException(
                    "Provider for " + type + " cannot be created", e);
```

```
            final FactoryConfigurationError error =
                    new FactoryConfigurationError(x, x.getMessage());
            throw error;
        }
    }

}
```

18.4 Location.java

Secuencia 193: javax/xml/stream/Location.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * Provides information on the location of an event.
 *
 * All the information provided by a Location is optional.  For example
 * an application may only report line numbers.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface Location {
  /**
```

```
   * Return the line number where the current event ends,
   * returns -1 if none is available.
   * @return the current line number
   */
  int getLineNumber();

  /**
   * Return the column number where the current event ends,
   * returns -1 if none is available.
   * @return the current column number
   */
  int getColumnNumber();

  /**
   * Return the byte or character offset into the input source this location
   * is pointing to. If the input source is a file or a byte stream then
   * this is the byte offset into that stream, but if the input source is
   * a character media then the offset is the character offset.
   * Returns -1 if there is no offset available.
   * @return the current offset
   */
  int getCharacterOffset();

  /**
   * Returns the public ID of the XML
   * @return the public ID, or null if not available
   */
  public String getPublicId();

  /**
   * Returns the system ID of the XML
   * @return the system ID, or null if not available
   */
  public String getSystemId();
}
```

18.5 SecuritySupport.java

Secuencia 194: javax/xml/stream/SecuritySupport.java

```
/*
 * Copyright (c) 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.stream;

import java.security.*;
import java.net.*;
import java.io.*;
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


    ClassLoader getContextClassLoader() throws SecurityException{
        return (ClassLoader)
                AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                //try {
                cl = Thread.currentThread().getContextClassLoader();
                //} catch (SecurityException ex) { }

                if (cl == null)
                    cl = ClassLoader.getSystemClassLoader();

                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }
```

```

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                    final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

}
```

18.6 StreamFilter.java

Secuencia 195: javax/xml/stream/StreamFilter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * This interface declares a simple filter interface that one can
 * create to filter XMLStreamReaders
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface StreamFilter {

  /**
   * Tests whether the current state is part of this stream.  This method
   * will return true if this filter accepts this event and false otherwise.
   *
   * The method should not change the state of the reader when accepting
   * a state.
   *
   * @param reader the event to test
   * @return true if this filter accepts this event, false otherwise
   */
  public boolean accept(XMLStreamReader reader);
}
```

18.7 XMLEventFactory.java

Secuencia 196: javax/xml/stream/XMLEventFactory.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009, 2013, by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;
import java.util.Iterator;
import javax.xml.namespace.NamespaceContext;
import javax.xml.namespace.QName;
import javax.xml.stream.events.*;
/**
 * This interface defines a utility class for creating instances of
 * XMLEvents
 * @version 1.2
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.events.StartElement
 * @see javax.xml.stream.events.EndElement
 * @see javax.xml.stream.events.ProcessingInstruction
 * @see javax.xml.stream.events.Comment
 * @see javax.xml.stream.events.Characters
 * @see javax.xml.stream.events.StartDocument
 * @see javax.xml.stream.events.EndDocument
 * @see javax.xml.stream.events.DTD
 * @since 1.6
 */
public abstract class XMLEventFactory {
  protected XMLEventFactory(){}

    static final String JAXPFACTORYID = "javax.xml.stream.XMLEventFactory";
    static final String DEFAULIMPL = "com.sun.xml.internal.stream.events.XMLEventFactoryImpl";


  /**
   * Creates a new instance of the factory in exactly the same manner as the
```

```
 * {@link #newFactory()} method.
 * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
 */
public static XMLEventFactory newInstance()
  throws FactoryConfigurationError
{
  return FactoryFinder.find(XMLEventFactory.class, DEFAULIMPL);
}

/**
 * Create a new instance of the factory.
 * <p>
 * This static method creates a new factory instance.
 * This method uses the following ordered lookup procedure to determine
 * the XMLEventFactory implementation class to load:
 * </p>
 * <ul>
 * <li>
 *   Use the javax.xml.stream.XMLEventFactory system property.
 * </li>
 * <li>
 *   Use the properties file "lib/stax.properties" in the JRE directory.
 *     This configuration file is in standard java.util.Properties format
 *     and contains the fully qualified name of the implementation class
 *     with the key being the system property defined above.
 * </li>
 * <li>
 *   Use the service-provider loading facilities, defined by the
 *   {@link java.util.ServiceLoader} class, to attempt to locate and load an
 *   implementation of the service using the {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
 *   the service-provider loading facility will use the {@linkplain
 *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
 *   to attempt to load the service. If the context class
 *   loader is null, the {@linkplain
 *   ClassLoader#getSystemClassLoader() system class loader} will be used.
 * </li>
 * <li>
 *   Otherwise, the system-default implementation is returned.
 * </li>
 * </ul>
 * <p>
 *   Once an application has obtained a reference to a XMLEventFactory it
 *   can use the factory to configure and obtain stream instances.
 * </p>
 * <p>
 *   Note that this is a new method that replaces the deprecated newInstance() method.
 *     No changes in behavior are defined by this replacement method relative to
 *     the deprecated method.
 * </p>
 * @throws FactoryConfigurationError in case of {@linkplain
 *   java.util.ServiceConfigurationError service configuration error} or if
 *   the implementation is not available or cannot be instantiated.
```

```
 */
public static XMLEventFactory newFactory()
  throws FactoryConfigurationError
{
  return FactoryFinder.find(XMLEventFactory.class, DEFAULIMPL);
}

/**
 * Create a new instance of the factory
 *
 * @param factoryId             Name of the factory to find, same as
 *                              a property name
 * @param classLoader           classLoader to use
 * @return the factory implementation
 * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
 *
 * @deprecated  This method has been deprecated to maintain API consistency.
 *              All newInstance methods have been replaced with corresponding
 *              newFactory methods. The replacement {@link
 *              #newFactory(java.lang.String, java.lang.ClassLoader)}
 *              method defines no changes in behavior.
 */
public static XMLEventFactory newInstance(String factoryId,
        ClassLoader classLoader)
        throws FactoryConfigurationError {
    //do not fallback if given classloader can't find the class, throw exception
    return FactoryFinder.find(XMLEventFactory.class, factoryId, classLoader, null);
}

/**
 * Create a new instance of the factory.
 * If the classLoader argument is null, then the ContextClassLoader is used.
 * <p>
 * This method uses the following ordered lookup procedure to determine
 * the XMLEventFactory implementation class to load:
 * </p>
 * <ul>
 * <li>
 *   Use the value of the system property identified by {@code factoryId}.
 * </li>
 * <li>
 *   Use the properties file "lib/stax.properties" in the JRE directory.
 *     This configuration file is in standard java.util.Properties format
 *     and contains the fully qualified name of the implementation class
 *     with the key being the given {@code factoryId}.
 * </li>
 * <li>
 *   If {@code factoryId} is "javax.xml.stream.XMLEventFactory",
 *   use the service-provider loading facilities, defined by the
 *   {@link java.util.ServiceLoader} class, to attempt to {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class, java.lang.ClassLoader) locate and load}
 *   an implementation of the service using the specified {@code ClassLoader}.
 *   If {@code classLoader} is null, the {@linkplain
```

```
  *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism} will apply:
  *   That is, the service-provider loading facility will use the {@linkplain
  *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
  *   to attempt to load the service. If the context class
  *   loader is null, the {@linkplain
  *   ClassLoader#getSystemClassLoader() system class loader} will be used.
  * </li>
  * <li>
  *   Otherwise, throws a {@link FactoryConfigurationError}.
  * </li>
  * </ul>
  *
  * <p>
  * Note that this is a new method that replaces the deprecated
  *   {@link #newInstance(java.lang.String, java.lang.ClassLoader)
  *   newInstance(String factoryId, ClassLoader classLoader)} method.
  * No changes in behavior are defined by this replacement method relative
  * to the deprecated method.
  * </p>
  *
  * @apiNote The parameter factoryId defined here is inconsistent with that
  * of other JAXP factories where the first parameter is fully qualified
  * factory class name that provides implementation of the factory.
  *
  * @param factoryId             Name of the factory to find, same as
  *                              a property name
  * @param classLoader           classLoader to use
  * @return the factory implementation
  * @throws FactoryConfigurationError in case of {@linkplain
  *   java.util.ServiceConfigurationError service configuration error} or if
  *   the implementation is not available or cannot be instantiated.
  */
 public static XMLEventFactory newFactory(String factoryId,
                                          ClassLoader classLoader)
         throws FactoryConfigurationError {
     //do not fallback if given classloader can't find the class, throw exception
     return FactoryFinder.find(XMLEventFactory.class, factoryId, classLoader, null);
 }

/**
  * This method allows setting of the Location on each event that
  * is created by this factory.  The values are copied by value into
  * the events created by this factory.  To reset the location
  * information set the location to null.
  * @param location the location to set on each event created
  */
 public abstract void setLocation(Location location);

 /**
  * Create a new Attribute
  * @param prefix the prefix of this attribute, may not be null
  * @param namespaceURI the attribute value is set to this value, may not be null
  * @param localName the local name of the XML name of the attribute, localName cannot be null
```

```
 * @param value the attribute value to set, may not be null
 * @return the Attribute with specified values
 */
public abstract Attribute createAttribute(String prefix, String namespaceURI, String localName,
    String value);

/**
 * Create a new Attribute
 * @param localName the local name of the XML name of the attribute, localName cannot be null
 * @param value the attribute value to set, may not be null
 * @return the Attribute with specified values
 */
public abstract Attribute createAttribute(String localName, String value);

/**
 * Create a new Attribute
 * @param name the qualified name of the attribute, may not be null
 * @param value the attribute value to set, may not be null
 * @return the Attribute with specified values
 */
public abstract Attribute createAttribute(QName name, String value);

/**
 * Create a new default Namespace
 * @param namespaceURI the default namespace uri
 * @return the Namespace with the specified value
 */
public abstract Namespace createNamespace(String namespaceURI);

/**
 * Create a new Namespace
 * @param prefix the prefix of this namespace, may not be null
 * @param namespaceUri the attribute value is set to this value, may not be null
 * @return the Namespace with the specified values
 */
public abstract Namespace createNamespace(String prefix, String namespaceUri);

/**
 * Create a new StartElement.  Namespaces can be added to this StartElement
 * by passing in an Iterator that walks over a set of Namespace interfaces.
 * Attributes can be added to this StartElement by passing an iterator
 * that walks over a set of Attribute interfaces.
 *
 * @param name the qualified name of the attribute, may not be null
 * @param attributes an optional unordered set of objects that
 * implement Attribute to add to the new StartElement, may be null
 * @param namespaces an optional unordered set of objects that
 * implement Namespace to add to the new StartElement, may be null
 * @return an instance of the requested StartElement
 */
public abstract StartElement createStartElement(QName name,
                                                Iterator attributes,
                                                Iterator namespaces);
```

```java

 /**
  * Create a new StartElement.  This defaults the NamespaceContext to
  * an empty NamespaceContext.  Querying this event for its namespaces or
  * attributes will result in an empty iterator being returned.
  *
  * @param namespaceUri the uri of the QName of the new StartElement
  * @param localName the local name of the QName of the new StartElement
  * @param prefix the prefix of the QName of the new StartElement
  * @return an instance of the requested StartElement
  */
 public abstract StartElement createStartElement(String prefix,
                                                 String namespaceUri,
                                                 String localName);
 /**
  * Create a new StartElement.  Namespaces can be added to this StartElement
  * by passing in an Iterator that walks over a set of Namespace interfaces.
  * Attributes can be added to this StartElement by passing an iterator
  * that walks over a set of Attribute interfaces.
  *
  * @param namespaceUri the uri of the QName of the new StartElement
  * @param localName the local name of the QName of the new StartElement
  * @param prefix the prefix of the QName of the new StartElement
  * @param attributes an unordered set of objects that implement
  * Attribute to add to the new StartElement
  * @param namespaces an unordered set of objects that implement
  * Namespace to add to the new StartElement
  * @return an instance of the requested StartElement
  */
 public abstract StartElement createStartElement(String prefix,
                                                 String namespaceUri,
                                                 String localName,
                                                 Iterator attributes,
                                                 Iterator namespaces
                                                 );
 /**
  * Create a new StartElement.  Namespaces can be added to this StartElement
  * by passing in an Iterator that walks over a set of Namespace interfaces.
  * Attributes can be added to this StartElement by passing an iterator
  * that walks over a set of Attribute interfaces.
  *
  * @param namespaceUri the uri of the QName of the new StartElement
  * @param localName the local name of the QName of the new StartElement
  * @param prefix the prefix of the QName of the new StartElement
  * @param attributes an unordered set of objects that implement
  * Attribute to add to the new StartElement, may be null
  * @param namespaces an unordered set of objects that implement
  * Namespace to add to the new StartElement, may be null
  * @param context the namespace context of this element
  * @return an instance of the requested StartElement
  */
 public abstract StartElement createStartElement(String prefix,
                                                 String namespaceUri,
```

```
                                                String localName,
                                                Iterator attributes,
                                                Iterator namespaces,
                                                NamespaceContext context
                                                );

  /**
   * Create a new EndElement
   * @param name the qualified name of the EndElement
   * @param namespaces an optional unordered set of objects that
   * implement Namespace that have gone out of scope, may be null
   * @return an instance of the requested EndElement
   */
  public abstract EndElement createEndElement(QName name,
                                              Iterator namespaces);

  /**
   * Create a new EndElement
   * @param namespaceUri the uri of the QName of the new StartElement
   * @param localName the local name of the QName of the new StartElement
   * @param prefix the prefix of the QName of the new StartElement
   * @return an instance of the requested EndElement
   */
  public abstract EndElement createEndElement(String prefix,
                                              String namespaceUri,
                                              String localName);
  /**
   * Create a new EndElement
   * @param namespaceUri the uri of the QName of the new StartElement
   * @param localName the local name of the QName of the new StartElement
   * @param prefix the prefix of the QName of the new StartElement
   * @param namespaces an unordered set of objects that implement
   * Namespace that have gone out of scope, may be null
   * @return an instance of the requested EndElement
   */
  public abstract EndElement createEndElement(String prefix,
                                              String namespaceUri,
                                              String localName,
                                              Iterator namespaces);

  /**
   * Create a Characters event, this method does not check if the content
   * is all whitespace.  To create a space event use #createSpace(String)
   * @param content the string to create
   * @return a Characters event
   */
  public abstract Characters createCharacters(String content);

  /**
   * Create a Characters event with the CData flag set to true
   * @param content the string to create
   * @return a Characters event
   */
```

```java
public abstract Characters createCData(String content);

/**
 * Create a Characters event with the isSpace flag set to true
 * @param content the content of the space to create
 * @return a Characters event
 */
public abstract Characters createSpace(String content);
/**
 * Create an ignorable space
 * @param content the space to create
 * @return a Characters event
 */
public abstract Characters createIgnorableSpace(String content);

/**
 * Creates a new instance of a StartDocument event
 * @return a StartDocument event
 */
public abstract StartDocument createStartDocument();

/**
 * Creates a new instance of a StartDocument event
 *
 * @param encoding the encoding style
 * @param version the XML version
 * @param standalone the status of standalone may be set to "true" or "false"
 * @return a StartDocument event
 */
public abstract StartDocument createStartDocument(String encoding,
                                                  String version,
                                                  boolean standalone);

/**
 * Creates a new instance of a StartDocument event
 *
 * @param encoding the encoding style
 * @param version the XML version
 * @return a StartDocument event
 */
public abstract StartDocument createStartDocument(String encoding,
                                                  String version);

/**
 * Creates a new instance of a StartDocument event
 *
 * @param encoding the encoding style
 * @return a StartDocument event
 */
public abstract StartDocument createStartDocument(String encoding);

/**
 * Creates a new instance of an EndDocument event
```

```
   * @return an EndDocument event
   */
  public abstract EndDocument createEndDocument();

  /** Creates a new instance of a EntityReference event
   *
   * @param name The name of the reference
   * @param declaration the declaration for the event
   * @return an EntityReference event
   */
  public abstract EntityReference createEntityReference(String name,
                                                        EntityDeclaration declaration);
  /**
   * Create a comment
   * @param text The text of the comment
   * a Comment event
   */
  public abstract Comment createComment(String text);

  /**
   * Create a processing instruction
   * @param target The target of the processing instruction
   * @param data The text of the processing instruction
   * @return a ProcessingInstruction event
   */
  public abstract ProcessingInstruction createProcessingInstruction(String target,
                                                                    String data);

  /**
   * Create a document type definition event
   * This string contains the entire document type declaration that matches
   * the doctypedecl in the XML 1.0 specification
   * @param dtd the text of the document type definition
   * @return a DTD event
   */
  public abstract DTD createDTD(String dtd);
}
```

18.8 XMLEventReader.java

Secuencia 197: javax/xml/stream/XMLEventReader.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.stream.events.XMLEvent;

import java.util.Iterator;

/**
 *
 * This is the top level interface for parsing XML Events.  It provides
 * the ability to peek at the next event and returns configuration
 * information through the property interface.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLInputFactory
 * @see XMLEventWriter
 * @since 1.6
 */
public interface XMLEventReader extends Iterator {
  /**
   * Get the next XMLEvent
   * @see XMLEvent
   * @throws XMLStreamException if there is an error with the underlying XML.
   * @throws NoSuchElementException iteration has no more elements.
   */
  public XMLEvent nextEvent() throws XMLStreamException;

  /**
   * Check if there are more events.
   * Returns true if there are more events and false otherwise.
   * @return true if the event reader has more events, false otherwise
   */
  public boolean hasNext();

  /**
   * Check the next XMLEvent without reading it from the stream.
   * Returns null if the stream is at EOF or has no more XMLEvents.
```

```
   * A call to peek() will be equal to the next return of next().
   * @see XMLEvent
   * @throws XMLStreamException
   */
  public XMLEvent peek() throws XMLStreamException;

  /**
   * Reads the content of a text-only element. Precondition:
   * the current event is START_ELEMENT. Postcondition:
   * The current event is the corresponding END_ELEMENT.
   * @throws XMLStreamException if the current event is not a START_ELEMENT
   * or if a non text element is encountered
   */
  public String getElementText() throws XMLStreamException;

  /**
   * Skips any insignificant space events until a START_ELEMENT or
   * END_ELEMENT is reached. If anything other than space characters are
   * encountered, an exception is thrown. This method should
   * be used when processing element-only content because
   * the parser is not able to recognize ignorable whitespace if
   * the DTD is missing or not interpreted.
   * @throws XMLStreamException if anything other than space characters are encountered
   */
  public XMLEvent nextTag() throws XMLStreamException;

  /**
   * Get the value of a feature/property from the underlying implementation
   * @param name The name of the property
   * @return The value of the property
   * @throws IllegalArgumentException if the property is not supported
   */
  public Object getProperty(java.lang.String name) throws java.lang.IllegalArgumentException;

  /**
   * Frees any resources associated with this Reader.  This method does not close the
   * underlying input source.
   * @throws XMLStreamException if there are errors freeing associated resources
   */
  public void close() throws XMLStreamException;
}
```

18.9 XMLEventWriter.java

Secuencia 198: javax/xml/stream/XMLEventWriter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.stream.events.*;
import javax.xml.stream.util.XMLEventConsumer;
import javax.xml.namespace.NamespaceContext;

/**
 *
 * This is the top level interface for writing XML documents.
 *
 * Instances of this interface are not required to validate the
 * form of the XML.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLEventReader
 * @see javax.xml.stream.events.XMLEvent
 * @see javax.xml.stream.events.Characters
 * @see javax.xml.stream.events.ProcessingInstruction
 * @see javax.xml.stream.events.StartElement
 * @see javax.xml.stream.events.EndElement
 * @since 1.6
 */
public interface XMLEventWriter extends XMLEventConsumer {

  /**
   * Writes any cached events to the underlying output mechanism
   * @throws XMLStreamException
   */
  public void flush() throws XMLStreamException;

  /**
   * Frees any resources associated with this stream
```

```
  * @throws XMLStreamException
  */
 public void close() throws XMLStreamException;

 /**
  * Add an event to the output stream
  * Adding a START_ELEMENT will open a new namespace scope that
  * will be closed when the corresponding END_ELEMENT is written.
  * <table border="2" rules="all" cellpadding="4">
  *   <thead>
  *     <tr>
  *       <th align="center" colspan="2">
  *         Required and optional fields for events added to the writer
  *       </th>
  *     </tr>
  *   </thead>
  *   <tbody>
  *     <tr>
  *       <th>Event Type</th>
  *       <th>Required Fields</th>
  *       <th>Optional Fields</th>
  *       <th>Required Behavior</th>
  *     </tr>
  *     <tr>
  *       <td> START_ELEMENT  </td>
  *       <td> QName name </td>
  *       <td> namespaces , attributes </td>
  *       <td> A START_ELEMENT will be written by writing the name,
  *       namespaces, and attributes of the event in XML 1.0 valid
  *       syntax for START_ELEMENTs.
  *       The name is written by looking up the prefix for
  *       the namespace uri.  The writer can be configured to
  *       respect prefixes of QNames.  If the writer is respecting
  *       prefixes it must use the prefix set on the QName.  The
  *       default behavior is to lookup the value for the prefix
  *       on the EventWriter's internal namespace context.
  *       Each attribute (if any)
  *       is written using the behavior specified in the attribute
  *       section of this table.  Each namespace (if any) is written
  *       using the behavior specified in the namespace section of this
  *       table.
  *       </td>
  *     </tr>
  *     <tr>
  *       <td> END_ELEMENT  </td>
  *       <td> Qname name  </td>
  *       <td> None </td>
  *       <td> A well formed END_ELEMENT tag is written.
  *       The name is written by looking up the prefix for
  *       the namespace uri.  The writer can be configured to
  *       respect prefixes of QNames.  If the writer is respecting
  *       prefixes it must use the prefix set on the QName.  The
  *       default behavior is to lookup the value for the prefix
```

```
 *       on the EventWriter's internal namespace context.
 *       If the END_ELEMENT name does not match the START_ELEMENT
 *       name an XMLStreamException is thrown.
 *       </td>
 *     </tr>
 *     <tr>
 *       <td> ATTRIBUTE  </td>
 *       <td> QName name , String value </td>
 *       <td> QName type </td>
 *       <td> An attribute is written using the same algorithm
 *            to find the lexical form as used in START_ELEMENT.
 *            The default is to use double quotes to wrap attribute
 *            values and to escape any double quotes found in the
 *            value.  The type value is ignored.
 *       </td>
 *     </tr>
 *     <tr>
 *       <td> NAMESPACE  </td>
 *       <td> String prefix, String namespaceURI,
 *            boolean isDefaultNamespaceDeclaration
 *      </td>
 *       <td> None  </td>
 *       <td> A namespace declaration is written.  If the
 *            namespace is a default namespace declaration
 *            (isDefault is true) then xmlns="$namespaceURI"
 *            is written and the prefix is optional.  If
 *            isDefault is false, the prefix must be declared
 *            and the writer must prepend xmlns to the prefix
 *            and write out a standard prefix declaration.
 *      </td>
 *     </tr>
 *     <tr>
 *       <td> PROCESSING_INSTRUCTION  </td>
 *       <td>   None</td>
 *       <td>   String target, String data</td>
 *       <td>   The data does not need to be present and may be
 *              null.  Target is required and many not be null.
 *              The writer
 *              will write data section
 *              directly after the target,
 *              enclosed in appropriate XML 1.0 syntax
 *      </td>
 *     </tr>
 *     <tr>
 *       <td> COMMENT  </td>
 *       <td> None  </td>
 *       <td> String comment  </td>
 *       <td> If the comment is present (not null) it is written, otherwise an
 *            an empty comment is written
 *      </td>
 *     </tr>
 *     <tr>
 *       <td> START_DOCUMENT  </td>
```

```
 *       <td> None  </td>
 *       <td> String encoding , boolean standalone, String version  </td>
 *       <td> A START_DOCUMENT event is not required to be written to the
 *             stream.  If present the attributes are written inside
 *             the appropriate XML declaration syntax
 *      </td>
 *     </tr>
 *     <tr>
 *       <td> END_DOCUMENT  </td>
 *       <td> None </td>
 *       <td> None  </td>
 *       <td> Nothing is written to the output  </td>
 *     </tr>
 *     <tr>
 *       <td> DTD  </td>
 *       <td> String DocumentTypeDefinition  </td>
 *       <td> None  </td>
 *       <td> The DocumentTypeDefinition is written to the output  </td>
 *     </tr>
 *   </tbody>
 * </table>
 * @param event the event to be added
 * @throws XMLStreamException
 */
public void add(XMLEvent event) throws XMLStreamException;

/**
 * Adds an entire stream to an output stream,
 * calls next() on the inputStream argument until hasNext() returns false
 * This should be treated as a convenience method that will
 * perform the following loop over all the events in an
 * event reader and call add on each event.
 *
 * @param reader the event stream to add to the output
 * @throws XMLStreamException
 */

public void add(XMLEventReader reader) throws XMLStreamException;

/**
 * Gets the prefix the uri is bound to
 * @param uri the uri to look up
 * @throws XMLStreamException
 */
public String getPrefix(String uri) throws XMLStreamException;

/**
 * Sets the prefix the uri is bound to.  This prefix is bound
 * in the scope of the current START_ELEMENT / END_ELEMENT pair.
 * If this method is called before a START_ELEMENT has been written
 * the prefix is bound in the root scope.
 * @param prefix the prefix to bind to the uri
 * @param uri the uri to bind to the prefix
```

```
   * @throws XMLStreamException
   */
  public void setPrefix(String prefix, String uri) throws XMLStreamException;

  /**
   * Binds a URI to the default namespace
   * This URI is bound
   * in the scope of the current START_ELEMENT / END_ELEMENT pair.
   * If this method is called before a START_ELEMENT has been written
   * the uri is bound in the root scope.
   * @param uri the uri to bind to the default namespace
   * @throws XMLStreamException
   */
  public void setDefaultNamespace(String uri) throws XMLStreamException;

  /**
   * Sets the current namespace context for prefix and uri bindings.
   * This context becomes the root namespace context for writing and
   * will replace the current root namespace context.  Subsequent calls
   * to setPrefix and setDefaultNamespace will bind namespaces using
   * the context passed to the method as the root context for resolving
   * namespaces.
   * @param context the namespace context to use for this writer
   * @throws XMLStreamException
   */
  public void setNamespaceContext(NamespaceContext context)
    throws XMLStreamException;

  /**
   * Returns the current namespace context.
   * @return the current namespace context
   */
  public NamespaceContext getNamespaceContext();


}
```

18.10 XMLInputFactory.java

Secuencia 199: javax/xml/stream/XMLInputFactory.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009, 2013, by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.stream.util.XMLEventAllocator;
import javax.xml.transform.Source;

/**
 * Defines an abstract implementation of a factory for getting streams.
 *
 * The following table defines the standard properties of this specification.
 * Each property varies in the level of support required by each implementation.
 * The level of support required is described in the 'Required' column.
 *
 *   <table border="2" rules="all" cellpadding="4">
 *    <thead>
 *      <tr>
 *        <th align="center" colspan="5">
 *          Configuration parameters
 *        </th>
 *      </tr>
 *    </thead>
 *    <tbody>
 *      <tr>
 *        <th>Property Name</th>
 *        <th>Behavior</th>
 *        <th>Return type</th>
 *        <th>Default Value</th>
 *        <th>Required</th>
 *      </tr>
 * <tr><td>javax.xml.stream.isValidating</td><td>Turns on/off implementation specific DTD
     validation</td><td>Boolean</td><td>False</td><td>No</td></tr>
 * <tr><td>javax.xml.stream.isNamespaceAware</td><td>Turns on/off namespace processing for XML 1.0
     support</td><td>Boolean</td><td>True</td><td>True (required) / False (optional)</td></tr>
 * <tr><td>javax.xml.stream.isCoalescing</td><td>Requires the processor to coalesce adjacent
     character data</td><td>Boolean</td><td>False</td><td>Yes</td></tr>
 * <tr><td>javax.xml.stream.isReplacingEntityReferences</td><td>replace internal entity references
     with their replacement text and report them as characters</td><td>Boolean</td><td>True</td><td
     >Yes</td></tr>
 *<tr><td>javax.xml.stream.isSupportingExternalEntities</td><td>Resolve external parsed entities</
```

```
    td><td>Boolean</td><td>Unspecified</td><td>Yes</td></tr>
 *<tr><td>javax.xml.stream.supportDTD</td><td>Use this property to request processors that do not
    support DTDs</td><td>Boolean</td><td>True</td><td>Yes</td></tr>
 *<tr><td>javax.xml.stream.reporter</td><td>sets/gets the impl of the XMLReporter </td><td>javax.
    xml.stream.XMLReporter</td><td>Null</td><td>Yes</td></tr>
 *<tr><td>javax.xml.stream.resolver</td><td>sets/gets the impl of the XMLResolver interface</td><td
    >javax.xml.stream.XMLResolver</td><td>Null</td><td>Yes</td></tr>
 *<tr><td>javax.xml.stream.allocator</td><td>sets/gets the impl of the XMLEventAllocator interface
    </td><td>javax.xml.stream.util.XMLEventAllocator</td><td>Null</td><td>Yes</td></tr>
 *    </tbody>
 *  </table>
 *
 *
 * @version 1.2
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLOutputFactory
 * @see XMLEventReader
 * @see XMLStreamReader
 * @see EventFilter
 * @see XMLReporter
 * @see XMLResolver
 * @see javax.xml.stream.util.XMLEventAllocator
 * @since 1.6
 */

public abstract class XMLInputFactory {
  /**
   * The property used to turn on/off namespace support,
   * this is to support XML 1.0 documents,
   * only the true setting must be supported
   */
  public static final String IS_NAMESPACE_AWARE=
    "javax.xml.stream.isNamespaceAware";

  /**
   * The property used to turn on/off implementation specific validation
   */
  public static final String IS_VALIDATING=
    "javax.xml.stream.isValidating";

  /**
   * The property that requires the parser to coalesce adjacent character data sections
   */
  public static final String IS_COALESCING=
    "javax.xml.stream.isCoalescing";

  /**
   * Requires the parser to replace internal
   * entity references with their replacement
   * text and report them as characters
   */
  public static final String IS_REPLACING_ENTITY_REFERENCES=
    "javax.xml.stream.isReplacingEntityReferences";
```

```java

  /**
   *  The property that requires the parser to resolve external parsed entities
   */
  public static final String IS_SUPPORTING_EXTERNAL_ENTITIES=
    "javax.xml.stream.isSupportingExternalEntities";

  /**
   *  The property that requires the parser to support DTDs
   */
  public static final String SUPPORT_DTD=
    "javax.xml.stream.supportDTD";

  /**
   * The property used to
   * set/get the implementation of the XMLReporter interface
   */
  public static final String REPORTER=
    "javax.xml.stream.reporter";

  /**
   * The property used to set/get the implementation of the XMLResolver
   */
  public static final String RESOLVER=
    "javax.xml.stream.resolver";

  /**
   * The property used to set/get the implementation of the allocator
   */
  public static final String ALLOCATOR=
    "javax.xml.stream.allocator";

  static final String DEFAULIMPL = "com.sun.xml.internal.stream.XMLInputFactoryImpl";

  protected XMLInputFactory(){}

  /**
   * Creates a new instance of the factory in exactly the same manner as the
   * {@link #newFactory()} method.
   * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
   */
  public static XMLInputFactory newInstance()
    throws FactoryConfigurationError
  {
    return FactoryFinder.find(XMLInputFactory.class, DEFAULIMPL);
  }

  /**
   * Create a new instance of the factory.
   * <p>
   * This static method creates a new factory instance.
   * This method uses the following ordered lookup procedure to determine
   * the XMLInputFactory implementation class to load:
```

```
 * </p>
 * <ul>
 * <li>
 *   Use the javax.xml.stream.XMLInputFactory system property.
 * </li>
 * <li>
 *   Use the properties file "lib/stax.properties" in the JRE directory.
 *     This configuration file is in standard java.util.Properties format
 *     and contains the fully qualified name of the implementation class
 *     with the key being the system property defined above.
 * </li>
 * <li>
 *   Use the service-provider loading facilities, defined by the
 *   {@link java.util.ServiceLoader} class, to attempt to locate and load an
 *   implementation of the service using the {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
 *   the service-provider loading facility will use the {@linkplain
 *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
 *   to attempt to load the service. If the context class
 *   loader is null, the {@linkplain
 *   ClassLoader#getSystemClassLoader() system class loader} will be used.
 * </li>
 * <li>
 * Otherwise, the system-default implementation is returned.
 * </li>
 * </ul>
 * <p>
 *   Once an application has obtained a reference to a XMLInputFactory it
 *   can use the factory to configure and obtain stream instances.
 * </p>
 * <p>
 *   Note that this is a new method that replaces the deprecated newInstance() method.
 *     No changes in behavior are defined by this replacement method relative to
 *     the deprecated method.
 * </p>
 * @throws FactoryConfigurationError in case of {@linkplain
 *   java.util.ServiceConfigurationError service configuration error} or if
 *   the implementation is not available or cannot be instantiated.
 */
public static XMLInputFactory newFactory()
  throws FactoryConfigurationError
{
  return FactoryFinder.find(XMLInputFactory.class, DEFAULIMPL);
}

/**
 * Create a new instance of the factory
 *
 * @param factoryId             Name of the factory to find, same as
 *                              a property name
 * @param classLoader           classLoader to use
 * @return the factory implementation
 * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
```

```
 *
 * @deprecated  This method has been deprecated to maintain API consistency.
 *              All newInstance methods have been replaced with corresponding
 *              newFactory methods. The replacement {@link
 *              #newFactory(java.lang.String, java.lang.ClassLoader)} method
 *              defines no changes in behavior.
 */
public static XMLInputFactory newInstance(String factoryId,
        ClassLoader classLoader)
        throws FactoryConfigurationError {
    //do not fallback if given classloader can't find the class, throw exception
    return FactoryFinder.find(XMLInputFactory.class, factoryId, classLoader, null);
}

/**
 * Create a new instance of the factory.
 * If the classLoader argument is null, then the ContextClassLoader is used.
 * <p>
 * This method uses the following ordered lookup procedure to determine
 * the XMLInputFactory implementation class to load:
 * </p>
 * <ul>
 * <li>
 *   Use the value of the system property identified by {@code factoryId}.
 * </li>
 * <li>
 *   Use the properties file "lib/stax.properties" in the JRE directory.
 *     This configuration file is in standard java.util.Properties format
 *     and contains the fully qualified name of the implementation class
 *     with the key being the given {@code factoryId}.
 * </li>
 * <li>
 *   If {@code factoryId} is "javax.xml.stream.XMLInputFactory",
 *   use the service-provider loading facilities, defined by the
 *   {@link java.util.ServiceLoader} class, to attempt to {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class, java.lang.ClassLoader) locate and load}
 *   an implementation of the service using the specified {@code ClassLoader}.
 *   If {@code classLoader} is null, the {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism} will apply:
 *   That is, the service-provider loading facility will use the {@linkplain
 *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
 *   to attempt to load the service. If the context class
 *   loader is null, the {@linkplain
 *   ClassLoader#getSystemClassLoader() system class loader} will be used.
 * </li>
 * <li>
 *   Otherwise, throws a {@link FactoryConfigurationError}.
 * </li>
 * </ul>
 *
 * <p>
 * Note that this is a new method that replaces the deprecated
 *   {@link #newInstance(java.lang.String, java.lang.ClassLoader)
```

```
 *   newInstance(String factoryId, ClassLoader classLoader)} method.
 * No changes in behavior are defined by this replacement method relative
 * to the deprecated method.
 * </p>
 *
 * @apiNote The parameter factoryId defined here is inconsistent with that
 * of other JAXP factories where the first parameter is fully qualified
 * factory class name that provides implementation of the factory.
 *
 * @param factoryId             Name of the factory to find, same as
 *                              a property name
 * @param classLoader           classLoader to use
 * @return the factory implementation
 * @throws FactoryConfigurationError in case of {@linkplain
 *   java.util.ServiceConfigurationError service configuration error} or if
 *   the implementation is not available or cannot be instantiated.
 * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
 */
public static XMLInputFactory newFactory(String factoryId,
        ClassLoader classLoader)
        throws FactoryConfigurationError {
    //do not fallback if given classloader can't find the class, throw exception
    return FactoryFinder.find(XMLInputFactory.class, factoryId, classLoader, null);
}

/**
 * Create a new XMLStreamReader from a reader
 * @param reader the XML data to read from
 * @throws XMLStreamException
 */
public abstract XMLStreamReader createXMLStreamReader(java.io.Reader reader)
  throws XMLStreamException;

/**
 * Create a new XMLStreamReader from a JAXP source.  This method is optional.
 * @param source the source to read from
 * @throws UnsupportedOperationException if this method is not
 * supported by this XMLInputFactory
 * @throws XMLStreamException
 */
public abstract XMLStreamReader createXMLStreamReader(Source source)
  throws XMLStreamException;

/**
 * Create a new XMLStreamReader from a java.io.InputStream
 * @param stream the InputStream to read from
 * @throws XMLStreamException
 */
public abstract XMLStreamReader createXMLStreamReader(java.io.InputStream stream)
  throws XMLStreamException;

/**
 * Create a new XMLStreamReader from a java.io.InputStream
```

```
 * @param stream the InputStream to read from
 * @param encoding the character encoding of the stream
 * @throws XMLStreamException
 */
public abstract XMLStreamReader createXMLStreamReader(java.io.InputStream stream, String encoding
    )
  throws XMLStreamException;

/**
 * Create a new XMLStreamReader from a java.io.InputStream
 * @param systemId the system ID of the stream
 * @param stream the InputStream to read from
 */
public abstract XMLStreamReader createXMLStreamReader(String systemId, java.io.InputStream stream
    )
  throws XMLStreamException;

/**
 * Create a new XMLStreamReader from a java.io.InputStream
 * @param systemId the system ID of the stream
 * @param reader the InputStream to read from
 */
public abstract XMLStreamReader createXMLStreamReader(String systemId, java.io.Reader reader)
  throws XMLStreamException;

/**
 * Create a new XMLEventReader from a reader
 * @param reader the XML data to read from
 * @throws XMLStreamException
 */
public abstract XMLEventReader createXMLEventReader(java.io.Reader reader)
  throws XMLStreamException;

/**
 * Create a new XMLEventReader from a reader
 * @param systemId the system ID of the input
 * @param reader the XML data to read from
 * @throws XMLStreamException
 */
public abstract XMLEventReader createXMLEventReader(String systemId, java.io.Reader reader)
  throws XMLStreamException;

/**
 * Create a new XMLEventReader from an XMLStreamReader.  After being used
 * to construct the XMLEventReader instance returned from this method
 * the XMLStreamReader must not be used.
 * @param reader the XMLStreamReader to read from (may not be modified)
 * @return a new XMLEventReader
 * @throws XMLStreamException
 */
public abstract XMLEventReader createXMLEventReader(XMLStreamReader reader)
  throws XMLStreamException;

```

```java
  /**
   * Create a new XMLEventReader from a JAXP source.
   * Support of this method is optional.
   * @param source the source to read from
   * @throws UnsupportedOperationException if this method is not
   * supported by this XMLInputFactory
   */
  public abstract XMLEventReader createXMLEventReader(Source source)
    throws XMLStreamException;

  /**
   * Create a new XMLEventReader from a java.io.InputStream
   * @param stream the InputStream to read from
   * @throws XMLStreamException
   */
  public abstract XMLEventReader createXMLEventReader(java.io.InputStream stream)
    throws XMLStreamException;

  /**
   * Create a new XMLEventReader from a java.io.InputStream
   * @param stream the InputStream to read from
   * @param encoding the character encoding of the stream
   * @throws XMLStreamException
   */
  public abstract XMLEventReader createXMLEventReader(java.io.InputStream stream, String encoding)
    throws XMLStreamException;

  /**
   * Create a new XMLEventReader from a java.io.InputStream
   * @param systemId the system ID of the stream
   * @param stream the InputStream to read from
   * @throws XMLStreamException
   */
  public abstract XMLEventReader createXMLEventReader(String systemId, java.io.InputStream stream)
    throws XMLStreamException;

  /**
   * Create a filtered reader that wraps the filter around the reader
   * @param reader the reader to filter
   * @param filter the filter to apply to the reader
   * @throws XMLStreamException
   */
  public abstract XMLStreamReader createFilteredReader(XMLStreamReader reader, StreamFilter filter)
    throws XMLStreamException;

  /**
   * Create a filtered event reader that wraps the filter around the event reader
   * @param reader the event reader to wrap
   * @param filter the filter to apply to the event reader
   * @throws XMLStreamException
   */
  public abstract XMLEventReader createFilteredReader(XMLEventReader reader, EventFilter filter)
    throws XMLStreamException;
```

```java

 /**
  * The resolver that will be set on any XMLStreamReader or XMLEventReader created
  * by this factory instance.
  */
 public abstract XMLResolver getXMLResolver();

 /**
  * The resolver that will be set on any XMLStreamReader or XMLEventReader created
  * by this factory instance.
  * @param resolver the resolver to use to resolve references
  */
 public abstract void  setXMLResolver(XMLResolver resolver);

 /**
  * The reporter that will be set on any XMLStreamReader or XMLEventReader created
  * by this factory instance.
  */
 public abstract XMLReporter getXMLReporter();

 /**
  * The reporter that will be set on any XMLStreamReader or XMLEventReader created
  * by this factory instance.
  * @param reporter the resolver to use to report non fatal errors
  */
 public abstract void setXMLReporter(XMLReporter reporter);

 /**
  * Allows the user to set specific feature/property on the underlying
  * implementation. The underlying implementation is not required to support
  * every setting of every property in the specification and may use
  * IllegalArgumentException to signal that an unsupported property may not be
  * set with the specified value.
  * <p>
  * All implementations that implement JAXP 1.5 or newer are required to
  * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property.
  * </p>
  * <ul>
  *   <li>
  *        <p>
  *        Access to external DTDs, external Entity References is restricted to the
  *        protocols specified by the property. If access is denied during parsing
  *        due to the restriction of this property, {@link javax.xml.stream.XMLStreamException}
  *        will be thrown by the {@link javax.xml.stream.XMLStreamReader#next()} or
  *        {@link javax.xml.stream.XMLEventReader#nextEvent()} method.
  *        </p>
  *   </li>
  * </ul>
  * @param name The name of the property (may not be null)
  * @param value The value of the property
  * @throws java.lang.IllegalArgumentException if the property is not supported
  */
 public abstract void setProperty(java.lang.String name, Object value)
```

```
    throws java.lang.IllegalArgumentException;

  /**
   * Get the value of a feature/property from the underlying implementation
   * @param name The name of the property (may not be null)
   * @return The value of the property
   * @throws IllegalArgumentException if the property is not supported
   */
  public abstract Object getProperty(java.lang.String name)
    throws java.lang.IllegalArgumentException;


  /**
   * Query the set of properties that this factory supports.
   *
   * @param name The name of the property (may not be null)
   * @return true if the property is supported and false otherwise
   */
  public abstract boolean isPropertySupported(String name);

  /**
   * Set a user defined event allocator for events
   * @param allocator the user defined allocator
   */
  public abstract void setEventAllocator(XMLEventAllocator allocator);

  /**
   * Gets the allocator used by streams created with this factory
   */
  public abstract XMLEventAllocator getEventAllocator();

}
```

18.11 XMLOutputFactory.java

Secuencia 200: javax/xml/stream/XMLOutputFactory.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009, 2013, by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.transform.Result;

/**
 * Defines an abstract implementation of a factory for
 * getting XMLEventWriters and XMLStreamWriters.
 *
 * The following table defines the standard properties of this specification.
 * Each property varies in the level of support required by each implementation.
 * The level of support required is described in the 'Required' column.
 *
 *     <table border="2" rules="all" cellpadding="4">
 *     <thead>
 *      <tr>
 *        <th align="center" colspan="2">
 *          Configuration parameters
 *        </th>
 *      </tr>
 *    </thead>
 *    <tbody>
 *      <tr>
 *        <th>Property Name</th>
 *        <th>Behavior</th>
 *        <th>Return type</th>
 *        <th>Default Value</th>
 *        <th>Required</th>
 *              </tr>
 *         <tr><td>javax.xml.stream.isRepairingNamespaces</td><td>defaults prefixes on the output
     side</td><td>Boolean</td><td>False</td><td>Yes</td></tr>
 *      </tbody>
 *   </table>
 *
 * <p>The following paragraphs describe the namespace and prefix repair algorithm:</p>
 *
 * <p>The property can be set with the following code line:
 * <code>setProperty("javax.xml.stream.isRepairingNamespaces",new Boolean(true|false));</code></p>
 *
 * <p>This property specifies that the writer default namespace prefix declarations.
 * The default value is false. </p>
 *
 * <p>If a writer isRepairingNamespaces it will create a namespace declaration
```

```
 * on the current StartElement for
 * any attribute that does not
 * currently have a namespace declaration in scope.  If the StartElement
 * has a uri but no prefix specified a prefix will be assigned, if the prefix
 * has not been declared in a parent of the current StartElement it will be declared
 * on the current StartElement.  If the defaultNamespace is bound and in scope
 * and the default namespace matches the URI of the attribute or StartElement
 * QName no prefix will be assigned.</p>
 *
 * <p>If an element or attribute name has a prefix, but is not
 * bound to any namespace URI, then the prefix will be removed
 * during serialization.</p>
 *
 * <p>If element and/or attribute names in the same start or
 * empty-element tag are bound to different namespace URIs and
 * are using the same prefix then the element or the first
 * occurring attribute retains the original prefix and the
 * following attributes have their prefixes replaced with a
 * new prefix that is bound to the namespace URIs of those
 * attributes. </p>
 *
 * <p>If an element or attribute name uses a prefix that is
 * bound to a different URI than that inherited from the
 * namespace context of the parent of that element and there
 * is no namespace declaration in the context of the current
 * element then such a namespace declaration is added. </p>
 *
 * <p>If an element or attribute name is bound to a prefix and
 * there is a namespace declaration that binds that prefix
 * to a different URI then that namespace declaration is
 * either removed if the correct mapping is inherited from
 * the parent context of that element, or changed to the
 * namespace URI of the element or attribute using that prefix.</p>
 *
 * @version 1.2
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLInputFactory
 * @see XMLEventWriter
 * @see XMLStreamWriter
 * @since 1.6
 */
public abstract class XMLOutputFactory {
  /**
   * Property used to set prefix defaulting on the output side
   */
  public static final String IS_REPAIRING_NAMESPACES=
    "javax.xml.stream.isRepairingNamespaces";

  static final String DEFAULIMPL = "com.sun.xml.internal.stream.XMLOutputFactoryImpl";

  protected XMLOutputFactory(){}

  /**
```

```
   * Creates a new instance of the factory in exactly the same manner as the
   * {@link #newFactory()} method.
   * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
   */
  public static XMLOutputFactory newInstance()
    throws FactoryConfigurationError
  {
    return FactoryFinder.find(XMLOutputFactory.class, DEFAULIMPL);
  }

  /**
   * Create a new instance of the factory.
   * <p>
   * This static method creates a new factory instance. This method uses the
   * following ordered lookup procedure to determine the XMLOutputFactory
   * implementation class to load:
   * </p>
   * <ul>
   * <li>
   *   Use the javax.xml.stream.XMLOutputFactory system property.
   * </li>
   * <li>
   *   Use the properties file "lib/stax.properties" in the JRE directory.
   *     This configuration file is in standard java.util.Properties format
   *     and contains the fully qualified name of the implementation class
   *     with the key being the system property defined above.
   * </li>
   * <li>
   *   Use the service-provider loading facilities, defined by the
   *   {@link java.util.ServiceLoader} class, to attempt to locate and load an
   *   implementation of the service using the {@linkplain
   *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
   *   the service-provider loading facility will use the {@linkplain
   *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
   *   to attempt to load the service. If the context class
   *   loader is null, the {@linkplain
   *   ClassLoader#getSystemClassLoader() system class loader} will be used.
   * </li>
   * <li>
   *   Otherwise, the system-default implementation is returned.
   * </li>
   * <p>
   * Once an application has obtained a reference to a XMLOutputFactory it
   * can use the factory to configure and obtain stream instances.
   * </p>
   * <p>
   * Note that this is a new method that replaces the deprecated newInstance() method.
   *   No changes in behavior are defined by this replacement method relative to the
   *   deprecated method.
   * </p>
   * @throws FactoryConfigurationError in case of {@linkplain
   *   java.util.ServiceConfigurationError service configuration error} or if
   *   the implementation is not available or cannot be instantiated.
```

```
 */
public static XMLOutputFactory newFactory()
  throws FactoryConfigurationError
{
  return FactoryFinder.find(XMLOutputFactory.class, DEFAULIMPL);
}

/**
 * Create a new instance of the factory.
 *
 * @param factoryId             Name of the factory to find, same as
 *                              a property name
 * @param classLoader           classLoader to use
 * @return the factory implementation
 * @throws FactoryConfigurationError if an instance of this factory cannot be loaded
 *
 * @deprecated  This method has been deprecated because it returns an
 *              instance of XMLInputFactory, which is of the wrong class.
 *              Use the new method {@link #newFactory(java.lang.String,
 *              java.lang.ClassLoader)} instead.
 */
public static XMLInputFactory newInstance(String factoryId,
        ClassLoader classLoader)
        throws FactoryConfigurationError {
    //do not fallback if given classloader can't find the class, throw exception
    return FactoryFinder.find(XMLInputFactory.class, factoryId, classLoader, null);
}

/**
 * Create a new instance of the factory.
 * If the classLoader argument is null, then the ContextClassLoader is used.
 * <p>
 * This method uses the following ordered lookup procedure to determine
 * the XMLOutputFactory implementation class to load:
 * </p>
 * <ul>
 * <li>
 *   Use the value of the system property identified by {@code factoryId}.
 * </li>
 * <li>
 *   Use the properties file "lib/stax.properties" in the JRE directory.
 *     This configuration file is in standard java.util.Properties format
 *     and contains the fully qualified name of the implementation class
 *     with the key being the given {@code factoryId}.
 * </li>
 * <li>
 *   If {@code factoryId} is "javax.xml.stream.XMLOutputFactory",
 *   use the service-provider loading facilities, defined by the
 *   {@link java.util.ServiceLoader} class, to attempt to {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class, java.lang.ClassLoader) locate and load}
 *   an implementation of the service using the specified {@code ClassLoader}.
 *   If {@code classLoader} is null, the {@linkplain
 *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism} will apply:
```

```
 *   That is, the service-provider loading facility will use the {@linkplain
 *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
 *   to attempt to load the service. If the context class
 *   loader is null, the {@linkplain
 *   ClassLoader#getSystemClassLoader() system class loader} will be used.
 * </li>
 * <li>
 *   Otherwise, throws a {@link FactoryConfigurationError}.
 * </li>
 * </ul>
 *
 * @apiNote The parameter factoryId defined here is inconsistent with that
 * of other JAXP factories where the first parameter is fully qualified
 * factory class name that provides implementation of the factory.
 *
 * <p>
 *   Note that this is a new method that replaces the deprecated
 *   {@link #newInstance(java.lang.String, java.lang.ClassLoader)
 *   newInstance(String factoryId, ClassLoader classLoader)} method.
 *   The original method was incorrectly defined to return XMLInputFactory.
 * </p>
 *
 * @param factoryId             Name of the factory to find, same as
 *                              a property name
 * @param classLoader           classLoader to use
 * @return the factory implementation
 * @throws FactoryConfigurationError in case of {@linkplain
 *   java.util.ServiceConfigurationError service configuration error} or if
 *   the implementation is not available or cannot be instantiated.
 */
public static XMLOutputFactory newFactory(String factoryId,
        ClassLoader classLoader)
        throws FactoryConfigurationError {
    //do not fallback if given classloader can't find the class, throw exception
    return FactoryFinder.find(XMLOutputFactory.class, factoryId, classLoader, null);
}

/**
 * Create a new XMLStreamWriter that writes to a writer
 * @param stream the writer to write to
 * @throws XMLStreamException
 */
public abstract XMLStreamWriter createXMLStreamWriter(java.io.Writer stream) throws
    XMLStreamException;

/**
 * Create a new XMLStreamWriter that writes to a stream
 * @param stream the stream to write to
 * @throws XMLStreamException
 */
public abstract XMLStreamWriter createXMLStreamWriter(java.io.OutputStream stream) throws
    XMLStreamException;

```

```
 /**
  * Create a new XMLStreamWriter that writes to a stream
  * @param stream the stream to write to
  * @param encoding the encoding to use
  * @throws XMLStreamException
  */
 public abstract XMLStreamWriter createXMLStreamWriter(java.io.OutputStream stream,
                                        String encoding) throws XMLStreamException;

 /**
  * Create a new XMLStreamWriter that writes to a JAXP result.  This method is optional.
  * @param result the result to write to
  * @throws UnsupportedOperationException if this method is not
  * supported by this XMLOutputFactory
  * @throws XMLStreamException
  */
 public abstract XMLStreamWriter createXMLStreamWriter(Result result) throws XMLStreamException;


 /**
  * Create a new XMLEventWriter that writes to a JAXP result.  This method is optional.
  * @param result the result to write to
  * @throws UnsupportedOperationException if this method is not
  * supported by this XMLOutputFactory
  * @throws XMLStreamException
  */
 public abstract XMLEventWriter createXMLEventWriter(Result result) throws XMLStreamException;

 /**
  * Create a new XMLEventWriter that writes to a stream
  * @param stream the stream to write to
  * @throws XMLStreamException
  */
 public abstract XMLEventWriter createXMLEventWriter(java.io.OutputStream stream) throws
     XMLStreamException;



 /**
  * Create a new XMLEventWriter that writes to a stream
  * @param stream the stream to write to
  * @param encoding the encoding to use
  * @throws XMLStreamException
  */
 public abstract XMLEventWriter createXMLEventWriter(java.io.OutputStream stream,
                                                     String encoding) throws XMLStreamException;

 /**
  * Create a new XMLEventWriter that writes to a writer
  * @param stream the stream to write to
  * @throws XMLStreamException
  */
 public abstract XMLEventWriter createXMLEventWriter(java.io.Writer stream) throws
```

```
    XMLStreamException;

  /**
   * Allows the user to set specific features/properties on the underlying implementation.
   * @param name The name of the property
   * @param value The value of the property
   * @throws java.lang.IllegalArgumentException if the property is not supported
   */
  public abstract void setProperty(java.lang.String name,
                                   Object value)
    throws IllegalArgumentException;

  /**
   * Get a feature/property on the underlying implementation
   * @param name The name of the property
   * @return The value of the property
   * @throws java.lang.IllegalArgumentException if the property is not supported
   */
  public abstract Object getProperty(java.lang.String name)
    throws IllegalArgumentException;

  /**
   * Query the set of properties that this factory supports.
   *
   * @param name The name of the property (may not be null)
   * @return true if the property is supported and false otherwise
   */
  public abstract boolean isPropertySupported(String name);
}
```

18.12 XMLReporter.java

Secuencia 201: javax/xml/stream/XMLReporter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * This interface is used to report non-fatal errors.
 * Only warnings should be echoed through this interface.
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface XMLReporter {

    /**

     * Report the desired message in an application specific format.

     * Only warnings and non-fatal errors should be reported through

     * this interface.

     * Fatal errors should be thrown as XMLStreamException.

     *

     * @param message the error message

     * @param errorType an implementation defined error type

     * @param relatedInformation information related to the error, if available

     * @param location the location of the error, if available

     * @throws XMLStreamException

     */
    public void report(String message, String errorType, Object relatedInformation, Location
        location)
            throws XMLStreamException;
}
```

18.13 XMLResolver.java

Secuencia 202: javax/xml/stream/XMLResolver.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * This interface is used to resolve resources during an XML parse.  If an application wishes to
 * perform custom entity resolution it must register an instance of this interface with
 * the XMLInputFactory using the setXMLResolver method.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface XMLResolver {

  /**
   * Retrieves a resource.  This resource can be of the following three return types:
   * (1) java.io.InputStream (2) javax.xml.stream.XMLStreamReader (3) java.xml.stream.
       XMLEventReader.
   * If this method returns null the processor will attempt to resolve the entity using its
   * default mechanism.
   *
   * @param publicID The public identifier of the external entity being referenced, or null if none
        was supplied.
   * @param systemID The system identifier of the external entity being referenced.
   * @param baseURI  Absolute base URI associated with systemId.
   * @param namespace The namespace of the entity to resolve.
   * @return The resource requested or null.
   * @throws XMLStreamException if there was a failure attempting to resolve the resource.
   */
```

```
  public Object resolveEntity(String publicID,
                              String systemID,
                              String baseURI,
                              String namespace)
    throws XMLStreamException;
}
```

18.14 XMLStreamConstants.java

Secuencia 203: javax/xml/stream/XMLStreamConstants.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.stream;

/**
 * This interface declares the constants used in this API.
 * Numbers in the range 0 to 256 are reserved for the specification,
 * user defined events must use event codes outside that range.
 *
 * @since 1.6
 */

public interface XMLStreamConstants {
  /**
   * Indicates an event is a start element
   * @see javax.xml.stream.events.StartElement
   */
  public static final int START_ELEMENT=1;
  /**
   * Indicates an event is an end element
   * @see javax.xml.stream.events.EndElement
```

```
 */
public static final int END_ELEMENT=2;
/**
 * Indicates an event is a processing instruction
 * @see javax.xml.stream.events.ProcessingInstruction
 */
public static final int PROCESSING_INSTRUCTION=3;

/**
 * Indicates an event is characters
 * @see javax.xml.stream.events.Characters
 */
public static final int CHARACTERS=4;

/**
 * Indicates an event is a comment
 * @see javax.xml.stream.events.Comment
 */
public static final int COMMENT=5;

/**
 * The characters are white space
 * (see [XML], 2.10 "White Space Handling").
 * Events are only reported as SPACE if they are ignorable white
 * space.  Otherwise they are reported as CHARACTERS.
 * @see javax.xml.stream.events.Characters
 */
public static final int SPACE=6;

/**
 * Indicates an event is a start document
 * @see javax.xml.stream.events.StartDocument
 */
public static final int START_DOCUMENT=7;

/**
 * Indicates an event is an end document
 * @see javax.xml.stream.events.EndDocument
 */
public static final int END_DOCUMENT=8;

/**
 * Indicates an event is an entity reference
 * @see javax.xml.stream.events.EntityReference
 */
public static final int ENTITY_REFERENCE=9;

/**
 * Indicates an event is an attribute
 * @see javax.xml.stream.events.Attribute
 */
public static final int ATTRIBUTE=10;

```

```
  /**
   * Indicates an event is a DTD
   * @see javax.xml.stream.events.DTD
   */
  public static final int DTD=11;

  /**
   * Indicates an event is a CDATA section
   * @see javax.xml.stream.events.Characters
   */
  public static final int CDATA=12;

  /**
   * Indicates the event is a namespace declaration
   *
   * @see javax.xml.stream.events.Namespace
   */
  public static final int NAMESPACE=13;

  /**
   * Indicates a Notation
   * @see javax.xml.stream.events.NotationDeclaration
   */
  public static final int NOTATION_DECLARATION=14;

  /**
   * Indicates a Entity Declaration
   * @see javax.xml.stream.events.NotationDeclaration
   */
  public static final int ENTITY_DECLARATION=15;
}
```

18.15 XMLStreamException.java

Secuencia 204: javax/xml/stream/XMLStreamException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

/**
 * The base exception for unexpected processing errors.  This Exception
 * class is used to report well-formedness errors as well as unexpected
 * processing conditions.
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */

public class XMLStreamException extends Exception {

  protected Throwable nested;
  protected Location location;

  /**
   * Default constructor
   */
  public XMLStreamException(){
    super();
  }

  /**
   * Construct an exception with the assocated message.
   *
   * @param msg the message to report
   */
  public XMLStreamException(String msg) {
    super(msg);
  }

  /**
   * Construct an exception with the assocated exception
   *
   * @param th a nested exception
   */
  public XMLStreamException(Throwable th) {
      super(th);
    nested = th;
  }

  /**
```

```
 * Construct an exception with the assocated message and exception
 *
 * @param th a nested exception
 * @param msg the message to report
 */
public XMLStreamException(String msg, Throwable th) {
  super(msg, th);
  nested = th;
}

/**
 * Construct an exception with the assocated message, exception and location.
 *
 * @param th a nested exception
 * @param msg the message to report
 * @param location the location of the error
 */
public XMLStreamException(String msg, Location location, Throwable th) {
  super("ParseError at [row,col]:["+location.getLineNumber()+","+
        location.getColumnNumber()+"]\n"+
        "Message: "+msg);
  nested = th;
  this.location = location;
}

/**
 * Construct an exception with the assocated message, exception and location.
 *
 * @param msg the message to report
 * @param location the location of the error
 */
public XMLStreamException(String msg,
                          Location location) {
  super("ParseError at [row,col]:["+location.getLineNumber()+","+
        location.getColumnNumber()+"]\n"+
        "Message: "+msg);
  this.location = location;
}


/**
 * Gets the nested exception.
 *
 * @return Nested exception
 */
public Throwable getNestedException() {
  return nested;
}

/**
 * Gets the location of the exception
 *
 * @return the location of the exception, may be null if none is available
```

```
   */
  public Location getLocation() {
    return location;
  }

}
```

18.16 XMLStreamReader.java

Secuencia 205: javax/xml/stream/XMLStreamReader.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import java.io.Reader;
import javax.xml.namespace.NamespaceContext;
import javax.xml.namespace.QName;

/**
 *  The XMLStreamReader interface allows forward, read-only access to XML.
 *  It is designed to be the lowest level and most efficient way to
 *  read XML data.
 *
 * <p> The XMLStreamReader is designed to iterate over XML using
 * next() and hasNext().  The data can be accessed using methods such as getEventType(),
 * getNamespaceURI(), getLocalName() and getText();
 *
```

```
* <p> The <a href="#next()">next()</a> method causes the reader to read the next parse event.
* The next() method returns an integer which identifies the type of event just read.
* <p> The event type can be determined using <a href="#getEventType()">getEventType()</a>.
* <p> Parsing events are defined as the XML Declaration, a DTD,
* start tag, character data, white space, end tag, comment,
* or processing instruction.  An attribute or namespace event may be encountered
* at the root level of a document as the result of a query operation.
*
* <p>For XML 1.0 compliance an XML processor must pass the
* identifiers of declared unparsed entities, notation declarations and their
* associated identifiers to the application.  This information is
* provided through the property API on this interface.
* The following two properties allow access to this information:
* javax.xml.stream.notations and javax.xml.stream.entities.
* When the current event is a DTD the following call will return a
* list of Notations
*  <code>List l = (List) getProperty("javax.xml.stream.notations");</code>
* The following call will return a list of entity declarations:
* <code>List l = (List) getProperty("javax.xml.stream.entities");</code>
* These properties can only be accessed during a DTD event and
* are defined to return null if the information is not available.
*
* <p>The following table describes which methods are valid in what state.
* If a method is called in an invalid state the method will throw a
* java.lang.IllegalStateException.
*
* <table border="2" rules="all" cellpadding="4">
*   <thead>
*     <tr>
*       <th align="center" colspan="2">
*         Valid methods for each state
*       </th>
*     </tr>
*   </thead>
*   <tbody>
*     <tr>
*       <th>Event Type</th>
*       <th>Valid Methods</th>
*     </tr>
*     <tr>
*       <td> All States  </td>
*       <td> getProperty(), hasNext(), require(), close(),
*            getNamespaceURI(), isStartElement(),
*            isEndElement(), isCharacters(), isWhiteSpace(),
*            getNamespaceContext(), getEventType(),getLocation(),
*            hasText(), hasName()
*       </td>
*     </tr>
*     <tr>
*       <td> START_ELEMENT  </td>
*       <td> next(), getName(), getLocalName(), hasName(), getPrefix(),
*            getAttributeXXX(), isAttributeSpecified(),
*            getNamespaceXXX(),
```

```
*           getElementText(), nextTag()
*       </td>
*     </tr>
*       <td> ATTRIBUTE  </td>
*       <td> next(), nextTag()
*           getAttributeXXX(), isAttributeSpecified(),
*       </td>
*     </tr>
*     </tr>
*       <td> NAMESPACE  </td>
*       <td> next(), nextTag()
*           getNamespaceXXX()
*       </td>
*     </tr>
*     <tr>
*       <td> END_ELEMENT  </td>
*       <td> next(), getName(), getLocalName(), hasName(), getPrefix(),
*           getNamespaceXXX(), nextTag()
*      </td>
*     </tr>
*     <tr>
*       <td> CHARACTERS  </td>
*       <td> next(), getTextXXX(), nextTag() </td>
*     </tr>
*     <tr>
*       <td> CDATA  </td>
*       <td> next(), getTextXXX(), nextTag() </td>
*     </tr>
*     <tr>
*       <td> COMMENT  </td>
*       <td> next(), getTextXXX(), nextTag() </td>
*     </tr>
*     <tr>
*       <td> SPACE  </td>
*       <td> next(), getTextXXX(), nextTag() </td>
*     </tr>
*     <tr>
*       <td> START_DOCUMENT  </td>
*       <td> next(), getEncoding(), getVersion(), isStandalone(), standaloneSet(),
*           getCharacterEncodingScheme(), nextTag()</td>
*     </tr>
*     <tr>
*       <td> END_DOCUMENT  </td>
*       <td> close()</td>
*     </tr>
*     <tr>
*       <td> PROCESSING_INSTRUCTION  </td>
*       <td> next(), getPITarget(), getPIData(), nextTag() </td>
*     </tr>
*     <tr>
*       <td> ENTITY_REFERENCE  </td>
*       <td> next(), getLocalName(), getText(), nextTag() </td>
*     </tr>
```

```
 *     <tr>
 *       <td> DTD  </td>
 *       <td> next(), getText(), nextTag() </td>
 *     </tr>
 *   </tbody>
 *  </table>
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.events.XMLEvent
 * @see XMLInputFactory
 * @see XMLStreamWriter
 * @since 1.6
 */
public interface XMLStreamReader extends XMLStreamConstants {
  /**
   * Get the value of a feature/property from the underlying implementation
   * @param name The name of the property, may not be null
   * @return The value of the property
   * @throws IllegalArgumentException if name is null
   */
  public Object getProperty(java.lang.String name) throws java.lang.IllegalArgumentException;

  /**
   * Get next parsing event - a processor may return all contiguous
   * character data in a single chunk, or it may split it into several chunks.
   * If the property javax.xml.stream.isCoalescing is set to true
   * element content must be coalesced and only one CHARACTERS event
   * must be returned for contiguous element content or
   * CDATA Sections.
   *
   * By default entity references must be
   * expanded and reported transparently to the application.
   * An exception will be thrown if an entity reference cannot be expanded.
   * If element content is empty (i.e. content is "") then no CHARACTERS event will be reported.
   *
   * <p>Given the following XML:<br>
   * &lt;foo>&lt;!--description-->content text&lt;![CDATA[&lt;greeting>Hello&lt;/greeting>]]>other content&lt;/foo><br>
   * The behavior of calling next() when being on foo will be:<br>
   * 1- the comment (COMMENT)<br>
   * 2- then the characters section (CHARACTERS)<br>
   * 3- then the CDATA section (another CHARACTERS)<br>
   * 4- then the next characters section (another CHARACTERS)<br>
   * 5- then the END_ELEMENT<br>
   *
   * <p><b>NOTE:</b> empty element (such as &lt;tag/>) will be reported
   *  with  two separate events: START_ELEMENT, END_ELEMENT - This preserves
   *   parsing equivalency of empty element to &lt;tag>&lt;/tag>.
   *
   * This method will throw an IllegalStateException if it is called after hasNext() returns false.
   * @see javax.xml.stream.events.XMLEvent
   * @return the integer code corresponding to the current parse event
```

```
 * @throws NoSuchElementException if this is called when hasNext() returns false
 * @throws XMLStreamException  if there is an error processing the underlying XML source
 */
public int next() throws XMLStreamException;

/**
 * Test if the current event is of the given type and if the namespace and name match the current
 * namespace and name of the current event.  If the namespaceURI is null it is not checked for equality,
 * if the localName is null it is not checked for equality.
 * @param type the event type
 * @param namespaceURI the uri of the event, may be null
 * @param localName the localName of the event, may be null
 * @throws XMLStreamException if the required values are not matched.
 */
public void require(int type, String namespaceURI, String localName) throws XMLStreamException;

/**
 * Reads the content of a text-only element, an exception is thrown if this is
 * not a text-only element.
 * Regardless of value of javax.xml.stream.isCoalescing this method always returns coalesced content.
 * <br /> Precondition: the current event is START_ELEMENT.
 * <br /> Postcondition: the current event is the corresponding END_ELEMENT.
 *
 * <br />The method does the following (implementations are free to optimized
 * but must do equivalent processing):
 * <pre>
 * if(getEventType() != XMLStreamConstants.START_ELEMENT) {
 * throw new XMLStreamException(
 * "parser must be on START_ELEMENT to read next text", getLocation());
 * }
 * int eventType = next();
 * StringBuffer content = new StringBuffer();
 * while(eventType != XMLStreamConstants.END_ELEMENT ) {
 * if(eventType == XMLStreamConstants.CHARACTERS
 * || eventType == XMLStreamConstants.CDATA
 * || eventType == XMLStreamConstants.SPACE
 * || eventType == XMLStreamConstants.ENTITY_REFERENCE) {
 * buf.append(getText());
 * } else if(eventType == XMLStreamConstants.PROCESSING_INSTRUCTION
 * || eventType == XMLStreamConstants.COMMENT) {
 * // skipping
 * } else if(eventType == XMLStreamConstants.END_DOCUMENT) {
 * throw new XMLStreamException(
 * "unexpected end of document when reading element text content", this);
 * } else if(eventType == XMLStreamConstants.START_ELEMENT) {
 * throw new XMLStreamException(
 * "element text content may not contain START_ELEMENT", getLocation());
 * } else {
 * throw new XMLStreamException(
 * "Unexpected event type "+eventType, getLocation());
 * }
```

```
 * eventType = next();
 * }
 * return buf.toString();
 * </pre>
 *
 * @throws XMLStreamException if the current event is not a START_ELEMENT
 * or if a non text element is encountered
 */
public String getElementText() throws XMLStreamException;

/**
 * Skips any white space (isWhiteSpace() returns true), COMMENT,
 * or PROCESSING_INSTRUCTION,
 * until a START_ELEMENT or END_ELEMENT is reached.
 * If other than white space characters, COMMENT, PROCESSING_INSTRUCTION, START_ELEMENT,
    END_ELEMENT
 * are encountered, an exception is thrown. This method should
 * be used when processing element-only content seperated by white space.
 *
 * <br /> Precondition: none
 * <br /> Postcondition: the current event is START_ELEMENT or END_ELEMENT
 * and cursor may have moved over any whitespace event.
 *
 * <br />Essentially it does the following (implementations are free to optimized
 * but must do equivalent processing):
 * <pre>
 * int eventType = next();
 * while((eventType == XMLStreamConstants.CHARACTERS &amp;&amp; isWhiteSpace()) // skip
    whitespace
 * || (eventType == XMLStreamConstants.CDATA &amp;&amp; isWhiteSpace())
 * // skip whitespace
 * || eventType == XMLStreamConstants.SPACE
 * || eventType == XMLStreamConstants.PROCESSING_INSTRUCTION
 * || eventType == XMLStreamConstants.COMMENT
 * ) {
 * eventType = next();
 * }
 * if (eventType != XMLStreamConstants.START_ELEMENT &amp;&amp; eventType != XMLStreamConstants.
    END_ELEMENT) {
 * throw new String XMLStreamException("expected start or end tag", getLocation());
 * }
 * return eventType;
 * </pre>
 *
 * @return the event type of the element read (START_ELEMENT or END_ELEMENT)
 * @throws XMLStreamException if the current event is not white space, PROCESSING_INSTRUCTION,
 * START_ELEMENT or END_ELEMENT
 * @throws NoSuchElementException if this is called when hasNext() returns false
 */
public int nextTag() throws XMLStreamException;

/**
 * Returns true if there are more parsing events and false
```

```
 * if there are no more events.  This method will return
 * false if the current state of the XMLStreamReader is
 * END_DOCUMENT
 * @return true if there are more events, false otherwise
 * @throws XMLStreamException if there is a fatal error detecting the next state
 */
public boolean hasNext() throws XMLStreamException;

/**
 * Frees any resources associated with this Reader.  This method does not close the
 * underlying input source.
 * @throws XMLStreamException if there are errors freeing associated resources
 */
public void close() throws XMLStreamException;

/**
 * Return the uri for the given prefix.
 * The uri returned depends on the current state of the processor.
 *
 * <p><strong>NOTE:</strong>The 'xml' prefix is bound as defined in
 * <a href="http://www.w3.org/TR/REC-xml-names/#ns-using">Namespaces in XML</a>
 * specification to "http://www.w3.org/XML/1998/namespace".
 *
 * <p><strong>NOTE:</strong> The 'xmlns' prefix must be resolved to following namespace
 * <a href="http://www.w3.org/2000/xmlns/">http://www.w3.org/2000/xmlns/</a>
 * @param prefix The prefix to lookup, may not be null
 * @return the uri bound to the given prefix or null if it is not bound
 * @throws IllegalArgumentException if the prefix is null
 */
public String getNamespaceURI(String prefix);

/**
 * Returns true if the cursor points to a start tag (otherwise false)
 * @return true if the cursor points to a start tag, false otherwise
 */
public boolean isStartElement();

/**
 * Returns true if the cursor points to an end tag (otherwise false)
 * @return true if the cursor points to an end tag, false otherwise
 */
public boolean isEndElement();

/**
 * Returns true if the cursor points to a character data event
 * @return true if the cursor points to character data, false otherwise
 */
public boolean isCharacters();

/**
 * Returns true if the cursor points to a character data event
 * that consists of all whitespace
 * @return true if the cursor points to all whitespace, false otherwise
```

```
 */
public boolean isWhiteSpace();


/**
 * Returns the normalized attribute value of the
 * attribute with the namespace and localName
 * If the namespaceURI is null the namespace
 * is not checked for equality
 * @param namespaceURI the namespace of the attribute
 * @param localName the local name of the attribute, cannot be null
 * @return returns the value of the attribute , returns null if not found
 * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
 */
public String getAttributeValue(String namespaceURI,
                                String localName);

/**
 * Returns the count of attributes on this START_ELEMENT,
 * this method is only valid on a START_ELEMENT or ATTRIBUTE.  This
 * count excludes namespace definitions.  Attribute indices are
 * zero-based.
 * @return returns the number of attributes
 * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
 */
public int getAttributeCount();

/** Returns the qname of the attribute at the provided index
 *
 * @param index the position of the attribute
 * @return the QName of the attribute
 * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
 */
public QName getAttributeName(int index);

/**
 * Returns the namespace of the attribute at the provided
 * index
 * @param index the position of the attribute
 * @return the namespace URI (can be null)
 * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
 */
public String getAttributeNamespace(int index);

/**
 * Returns the localName of the attribute at the provided
 * index
 * @param index the position of the attribute
 * @return the localName of the attribute
 * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
 */
public String getAttributeLocalName(int index);

```

```java
 /**
  * Returns the prefix of this attribute at the
  * provided index
  * @param index the position of the attribute
  * @return the prefix of the attribute
  * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
  */
 public String getAttributePrefix(int index);

 /**
  * Returns the XML type of the attribute at the provided
  * index
  * @param index the position of the attribute
  * @return the XML type of the attribute
  * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
  */
 public String getAttributeType(int index);

 /**
  * Returns the value of the attribute at the
  * index
  * @param index the position of the attribute
  * @return the attribute value
  * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
  */
 public String getAttributeValue(int index);

 /**
  * Returns a boolean which indicates if this
  * attribute was created by default
  * @param index the position of the attribute
  * @return true if this is a default attribute
  * @throws IllegalStateException if this is not a START_ELEMENT or ATTRIBUTE
  */
 public boolean isAttributeSpecified(int index);

 /**
  * Returns the count of namespaces declared on this START_ELEMENT or END_ELEMENT,
  * this method is only valid on a START_ELEMENT, END_ELEMENT or NAMESPACE. On
  * an END_ELEMENT the count is of the namespaces that are about to go
  * out of scope.  This is the equivalent of the information reported
  * by SAX callback for an end element event.
  * @return returns the number of namespace declarations on this specific element
  * @throws IllegalStateException if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE
  */
 public int getNamespaceCount();

 /**
  * Returns the prefix for the namespace declared at the
  * index.  Returns null if this is the default namespace
  * declaration
  *
  * @param index the position of the namespace declaration
```

```
 * @return returns the namespace prefix
 * @throws IllegalStateException if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE
 */
public String getNamespacePrefix(int index);

/**
 * Returns the uri for the namespace declared at the
 * index.
 *
 * @param index the position of the namespace declaration
 * @return returns the namespace uri
 * @throws IllegalStateException if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE
 */
public String getNamespaceURI(int index);

/**
 * Returns a read only namespace context for the current
 * position.  The context is transient and only valid until
 * a call to next() changes the state of the reader.
 * @return return a namespace context
 */
public NamespaceContext getNamespaceContext();

/**
 * Returns a reader that points to the current start element
 * and all of its contents.  Throws an XMLStreamException if the
 * cursor does not point to a START_ELEMENT.<p>
 * The sub stream is read from it MUST be read before the parent stream is
 * moved on, if not any call on the sub stream will cause an XMLStreamException to be
 * thrown.   The parent stream will always return the same result from next()
 * whatever is done to the sub stream.
 * @return an XMLStreamReader which points to the next element
 */
//  public XMLStreamReader subReader() throws XMLStreamException;

/**
 * Allows the implementation to reset and reuse any underlying tables
 */
//  public void recycle() throws XMLStreamException;

/**
 * Returns an integer code that indicates the type
 * of the event the cursor is pointing to.
 */
public int getEventType();

/**
 * Returns the current value of the parse event as a string,
 * this returns the string value of a CHARACTERS event,
 * returns the value of a COMMENT, the replacement value
 * for an ENTITY_REFERENCE, the string value of a CDATA section,
 * the string value for a SPACE event,
 * or the String value of the internal subset of the DTD.
```

```
 * If an ENTITY_REFERENCE has been resolved, any character data
 * will be reported as CHARACTERS events.
 * @return the current text or null
 * @throws java.lang.IllegalStateException if this state is not
 * a valid text state.
 */
public String getText();

/**
 * Returns an array which contains the characters from this event.
 * This array should be treated as read-only and transient. I.e. the array will
 * contain the text characters until the XMLStreamReader moves on to the next event.
 * Attempts to hold onto the character array beyond that time or modify the
 * contents of the array are breaches of the contract for this interface.
 * @return the current text or an empty array
 * @throws java.lang.IllegalStateException if this state is not
 * a valid text state.
 */
public char[] getTextCharacters();

/**
 * Gets the the text associated with a CHARACTERS, SPACE or CDATA event.
 * Text starting a "sourceStart" is copied into "target" starting at "targetStart".
 * Up to "length" characters are copied.  The number of characters actually copied is returned.
 *
 * The "sourceStart" argument must be greater or equal to 0 and less than or equal to
 * the number of characters associated with the event.  Usually, one requests text starting at a "sourceStart" of 0.
 * If the number of characters actually copied is less than the "length", then there is no more text.
 * Otherwise, subsequent calls need to be made until all text has been retrieved. For example:
 *
 *<code>
 * int length = 1024;
 * char[] myBuffer = new char[ length ];
 *
 * for ( int sourceStart = 0 ; ; sourceStart += length )
 * {
 *    int nCopied = stream.getTextCharacters( sourceStart, myBuffer, 0, length );
 *
 *   if (nCopied < length)
 *       break;
 * }
 * </code>
 * XMLStreamException may be thrown if there are any XML errors in the underlying source.
 * The "targetStart" argument must be greater than or equal to 0 and less than the length of "target",
 * Length must be greater than 0 and "targetStart + length" must be less than or equal to length of "target".
 *
 * @param sourceStart the index of the first character in the source array to copy
 * @param target the destination array
 * @param targetStart the start offset in the target array
```

```
   * @param length the number of characters to copy
   * @return the number of characters actually copied
   * @throws XMLStreamException if the underlying XML source is not well-formed
   * @throws IndexOutOfBoundsException if targetStart < 0 or > than the length of target
   * @throws IndexOutOfBoundsException if length < 0 or targetStart + length > length of target
   * @throws UnsupportedOperationException if this method is not supported
   * @throws NullPointerException is if target is null
   */
   public int getTextCharacters(int sourceStart, char[] target, int targetStart, int length)
     throws XMLStreamException;

  /**
   * Gets the text associated with a CHARACTERS, SPACE or CDATA event.  Allows the underlying
   * implementation to return the text as a stream of characters.  The reference to the
   * Reader returned by this method is only valid until next() is called.
   *
   * All characters must have been checked for well-formedness.
   *
   * <p> This method is optional and will throw UnsupportedOperationException if it is not
       supported.
   * @throws UnsupportedOperationException if this method is not supported
   * @throws IllegalStateException if this is not a valid text state
   */
  //public Reader getTextStream();

  /**
   * Returns the offset into the text character array where the first
   * character (of this text event) is stored.
   * @throws java.lang.IllegalStateException if this state is not
   * a valid text state.
   */
  public int getTextStart();

  /**
   * Returns the length of the sequence of characters for this
   * Text event within the text character array.
   * @throws java.lang.IllegalStateException if this state is not
   * a valid text state.
   */
  public int getTextLength();

  /**
   * Return input encoding if known or null if unknown.
   * @return the encoding of this instance or null
   */
  public String getEncoding();

  /**
   * Return true if the current event has text, false otherwise
   * The following events have text:
   * CHARACTERS,DTD ,ENTITY_REFERENCE, COMMENT, SPACE
   */
  public boolean hasText();
```

```java

/**
 * Return the current location of the processor.
 * If the Location is unknown the processor should return
 * an implementation of Location that returns -1 for the
 * location and null for the publicId and systemId.
 * The location information is only valid until next() is
 * called.
 */
public Location getLocation();

/**
 * Returns a QName for the current START_ELEMENT or END_ELEMENT event
 * @return the QName for the current START_ELEMENT or END_ELEMENT event
 * @throws IllegalStateException if this is not a START_ELEMENT or
 * END_ELEMENT
 */
public QName getName();

/**
 * Returns the (local) name of the current event.
 * For START_ELEMENT or END_ELEMENT returns the (local) name of the current element.
 * For ENTITY_REFERENCE it returns entity name.
 * The current event must be START_ELEMENT or END_ELEMENT,
 * or ENTITY_REFERENCE
 * @return the localName
 * @throws IllegalStateException if this not a START_ELEMENT,
 * END_ELEMENT or ENTITY_REFERENCE
 */
public String getLocalName();

/**
 * returns true if the current event has a name (is a START_ELEMENT or END_ELEMENT)
 * returns false otherwise
 */
public boolean hasName();

/**
 * If the current event is a START_ELEMENT or END_ELEMENT  this method
 * returns the URI of the prefix or the default namespace.
 * Returns null if the event does not have a prefix.
 * @return the URI bound to this elements prefix, the default namespace, or null
 */
public String getNamespaceURI();

/**
 * Returns the prefix of the current event or null if the event does not have a prefix
 * @return the prefix or null
 */
public String getPrefix();

/**
 * Get the xml version declared on the xml declaration
```

```
   * Returns null if none was declared
   * @return the XML version or null
   */
  public String getVersion();

  /**
   * Get the standalone declaration from the xml declaration
   * @return true if this is standalone, or false otherwise
   */
  public boolean isStandalone();

  /**
   * Checks if standalone was set in the document
   * @return true if standalone was set in the document, or false otherwise
   */
  public boolean standaloneSet();

  /**
   * Returns the character encoding declared on the xml declaration
   * Returns null if none was declared
   * @return the encoding declared in the document or null
   */
  public String getCharacterEncodingScheme();

  /**
   * Get the target of a processing instruction
   * @return the target or null
   */
  public String getPITarget();

  /**
   * Get the data section of a processing instruction
   * @return the data or null
   */
  public String getPIData();
}
```

18.17 XMLStreamWriter.java

Secuencia 206: javax/xml/stream/XMLStreamWriter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream;

import javax.xml.namespace.NamespaceContext;

/**
 * The XMLStreamWriter interface specifies how to write XML.  The XMLStreamWriter  does
 * not perform well formedness checking on its input.  However
 * the writeCharacters method is required to escape &amp; , &lt; and &gt;
 * For attribute values the writeAttribute method will escape the
 * above characters plus &quot; to ensure that all character content
 * and attribute values are well formed.
 *
 * Each NAMESPACE
 * and ATTRIBUTE must be individually written.
 *
 * <table border="1" cellpadding="2" cellspacing="0">
 *     <thead>
 *         <tr>
 *             <th colspan="5">XML Namespaces, <code>javax.xml.stream.isRepairingNamespaces</code>
     and write method behaviour</th>
 *         </tr>
 *         <tr>
 *             <th>Method</th> <!-- method -->
 *             <th colspan="2"><code>isRepairingNamespaces</code> == true</th>
 *             <th colspan="2"><code>isRepairingNamespaces</code> == false</th>
 *         </tr>
 *         <tr>
 *             <th></th> <!-- method -->
 *             <th>namespaceURI bound</th>
 *             <th>namespaceURI unbound</th>
 *             <th>namespaceURI bound</th>
 *             <th>namespaceURI unbound</th>
 *         </tr>
 *     </thead>
 *
 *     <tbody>
 *         <tr>
 *             <th><code>writeAttribute(namespaceURI, localName, value)</code></th>
```

```
*            <!-- isRepairingNamespaces == true -->
*            <td>
*                <!-- namespaceURI bound -->
*                prefix:localName="value" <sup>[1]</sup>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                xmlns:{generated}="namespaceURI" {generated}:localName="value"
*            </td>
*            <!-- isRepairingNamespaces == false -->
*            <td>
*                <!-- namespaceURI bound -->
*                prefix:localName="value" <sup>[1]</sup>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                <code>XMLStreamException</code>
*            </td>
*        </tr>
*
*        <tr>
*            <th><code>writeAttribute(prefix, namespaceURI, localName, value)</code></th>
*            <!-- isRepairingNamespaces == true -->
*            <td>
*                <!-- namespaceURI bound -->
*                bound to same prefix:<br />
*                prefix:localName="value" <sup>[1]</sup><br />
*                <br />
*                bound to different prefix:<br />
*                xmlns:{generated}="namespaceURI" {generated}:localName="value"
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                xmlns:prefix="namespaceURI" prefix:localName="value" <sup>[3]</sup>
*            </td>
*            <!-- isRepairingNamespaces == false -->
*            <td>
*                <!-- namespaceURI bound -->
*                bound to same prefix:<br />
*                prefix:localName="value" <sup>[1][2]</sup><br />
*                <br />
*                bound to different prefix:<br />
*                <code>XMLStreamException</code><sup>[2]</sup>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                xmlns:prefix="namespaceURI" prefix:localName="value" <sup>[2][5]</sup>
*            </td>
*        </tr>
*
*        <tr>
*            <th><code>writeStartElement(namespaceURI, localName)</code><br />
*                <br />
```

```
*                <code>writeEmptyElement(namespaceURI, localName)</code></th>
*            <!-- isRepairingNamespaces == true -->
*            <td >
*                <!-- namespaceURI bound -->
*                &lt;prefix:localName&gt; <sup>[1]</sup>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                &lt;{generated}:localName xmlns:{generated}="namespaceURI"&gt;
*            </td>
*            <!-- isRepairingNamespaces == false -->
*            <td>
*                <!-- namespaceURI bound -->
*                &lt;prefix:localName&gt; <sup>[1]</sup>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                <code>XMLStreamException</code>
*            </td>
*        </tr>
*
*        <tr>
*            <th><code>writeStartElement(prefix, localName, namespaceURI)</code><br />
*                <br />
*                <code>writeEmptyElement(prefix, localName, namespaceURI)</code></th>
*            <!-- isRepairingNamespaces == true -->
*            <td>
*                <!-- namespaceURI bound -->
*                bound to same prefix:<br />
*                &lt;prefix:localName&gt; <sup>[1]</sup><br />
*                <br />
*                bound to different prefix:<br />
*                &lt;{generated}:localName xmlns:{generated}="namespaceURI"&gt;
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                &lt;prefix:localName xmlns:prefix="namespaceURI"&gt; <sup>[4]</sup>
*            </td>
*            <!-- isRepairingNamespaces == false -->
*            <td>
*                <!-- namespaceURI bound -->
*                bound to same prefix:<br />
*                &lt;prefix:localName&gt; <sup>[1]</sup><br />
*                <br />
*                bound to different prefix:<br />
*                <code>XMLStreamException</code>
*            </td>
*            <td>
*                <!-- namespaceURI unbound -->
*                &lt;prefix:localName&gt; 
*            </td>
*        </tr>
*    </tbody>
```

```
 *     <tfoot>
 *         <tr>
 *             <td colspan="5">
 *                 Notes:
 *                 <ul>
 *                     <li>[1] if namespaceURI == default Namespace URI, then no prefix is written</li>
 *                     <li>[2] if prefix == "" || null && namespaceURI == "", then no prefix or Namespace declaration is generated or written</li>
 *                     <li>[3] if prefix == "" || null, then a prefix is randomly generated</li>
 *                     <li>[4] if prefix == "" || null, then it is treated as the default Namespace and no prefix is generated or written, an xmlns declaration is generated and written if the namespaceURI is unbound</li>
 *                     <li>[5] if prefix == "" || null, then it is treated as an invalid attempt to define the default Namespace and an XMLStreamException is thrown</li>
 *                 </ul>
 *             </td>
 *         </tr>
 *     </tfoot>
 * </table>
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLOutputFactory
 * @see XMLStreamReader
 * @since 1.6
 */
public interface XMLStreamWriter {

  /**
   * Writes a start tag to the output.  All writeStartElement methods
   * open a new scope in the internal namespace context.  Writing the
   * corresponding EndElement causes the scope to be closed.
   * @param localName local name of the tag, may not be null
   * @throws XMLStreamException
   */
  public void writeStartElement(String localName)
    throws XMLStreamException;

  /**
   * Writes a start tag to the output
   * @param namespaceURI the namespaceURI of the prefix to use, may not be null
   * @param localName local name of the tag, may not be null
   * @throws XMLStreamException if the namespace URI has not been bound to a prefix and
   * javax.xml.stream.isRepairingNamespaces has not been set to true
   */
  public void writeStartElement(String namespaceURI, String localName)
    throws XMLStreamException;

  /**
   * Writes a start tag to the output
   * @param localName local name of the tag, may not be null
   * @param prefix the prefix of the tag, may not be null
```

```
 * @param namespaceURI the uri to bind the prefix to, may not be null
 * @throws XMLStreamException
 */
public void writeStartElement(String prefix,
                              String localName,
                              String namespaceURI)
  throws XMLStreamException;

/**
 * Writes an empty element tag to the output
 * @param namespaceURI the uri to bind the tag to, may not be null
 * @param localName local name of the tag, may not be null
 * @throws XMLStreamException if the namespace URI has not been bound to a prefix and
 * javax.xml.stream.isRepairingNamespaces has not been set to true
 */
public void writeEmptyElement(String namespaceURI, String localName)
  throws XMLStreamException;

/**
 * Writes an empty element tag to the output
 * @param prefix the prefix of the tag, may not be null
 * @param localName local name of the tag, may not be null
 * @param namespaceURI the uri to bind the tag to, may not be null
 * @throws XMLStreamException
 */
public void writeEmptyElement(String prefix, String localName, String namespaceURI)
  throws XMLStreamException;

/**
 * Writes an empty element tag to the output
 * @param localName local name of the tag, may not be null
 * @throws XMLStreamException
 */
public void writeEmptyElement(String localName)
  throws XMLStreamException;

/**
 * Writes string data to the output without checking for well formedness.
 * The data is opaque to the XMLStreamWriter, i.e. the characters are written
 * blindly to the underlying output.  If the method cannot be supported
 * in the currrent writing context the implementation may throw a
 * UnsupportedOperationException.  For example note that any
 * namespace declarations, end tags, etc. will be ignored and could
 * interfere with proper maintanence of the writers internal state.
 *
 * @param data the data to write
 */
//  public void writeRaw(String data) throws XMLStreamException;

/**
 * Writes an end tag to the output relying on the internal
 * state of the writer to determine the prefix and local name
 * of the event.
```

```
 * @throws XMLStreamException
 */
public void writeEndElement()
  throws XMLStreamException;

/**
 * Closes any start tags and writes corresponding end tags.
 * @throws XMLStreamException
 */
public void writeEndDocument()
  throws XMLStreamException;

/**
 * Close this writer and free any resources associated with the
 * writer.  This must not close the underlying output stream.
 * @throws XMLStreamException
 */
public void close()
  throws XMLStreamException;

/**
 * Write any cached data to the underlying output mechanism.
 * @throws XMLStreamException
 */
public void flush()
  throws XMLStreamException;

/**
 * Writes an attribute to the output stream without
 * a prefix.
 * @param localName the local name of the attribute
 * @param value the value of the attribute
 * @throws IllegalStateException if the current state does not allow Attribute writing
 * @throws XMLStreamException
 */
public void writeAttribute(String localName, String value)
  throws XMLStreamException;

/**
 * Writes an attribute to the output stream
 * @param prefix the prefix for this attribute
 * @param namespaceURI the uri of the prefix for this attribute
 * @param localName the local name of the attribute
 * @param value the value of the attribute
 * @throws IllegalStateException if the current state does not allow Attribute writing
 * @throws XMLStreamException if the namespace URI has not been bound to a prefix and
 * javax.xml.stream.isRepairingNamespaces has not been set to true
 */

public void writeAttribute(String prefix,
                           String namespaceURI,
                           String localName,
                           String value)
```

```
    throws XMLStreamException;

  /**
   * Writes an attribute to the output stream
   * @param namespaceURI the uri of the prefix for this attribute
   * @param localName the local name of the attribute
   * @param value the value of the attribute
   * @throws IllegalStateException if the current state does not allow Attribute writing
   * @throws XMLStreamException if the namespace URI has not been bound to a prefix and
   * javax.xml.stream.isRepairingNamespaces has not been set to true
   */
  public void writeAttribute(String namespaceURI,
                             String localName,
                             String value)
    throws XMLStreamException;

  /**
   * Writes a namespace to the output stream
   * If the prefix argument to this method is the empty string,
   * "xmlns", or null this method will delegate to writeDefaultNamespace
   *
   * @param prefix the prefix to bind this namespace to
   * @param namespaceURI the uri to bind the prefix to
   * @throws IllegalStateException if the current state does not allow Namespace writing
   * @throws XMLStreamException
   */
  public void writeNamespace(String prefix, String namespaceURI)
    throws XMLStreamException;

  /**
   * Writes the default namespace to the stream
   * @param namespaceURI the uri to bind the default namespace to
   * @throws IllegalStateException if the current state does not allow Namespace writing
   * @throws XMLStreamException
   */
  public void writeDefaultNamespace(String namespaceURI)
    throws XMLStreamException;

  /**
   * Writes an xml comment with the data enclosed
   * @param data the data contained in the comment, may be null
   * @throws XMLStreamException
   */
  public void writeComment(String data)
    throws XMLStreamException;

  /**
   * Writes a processing instruction
   * @param target the target of the processing instruction, may not be null
   * @throws XMLStreamException
   */
  public void writeProcessingInstruction(String target)
    throws XMLStreamException;
```

```java

  /**
   * Writes a processing instruction
   * @param target the target of the processing instruction, may not be null
   * @param data the data contained in the processing instruction, may not be null
   * @throws XMLStreamException
   */
  public void writeProcessingInstruction(String target,
                                         String data)
    throws XMLStreamException;

  /**
   * Writes a CData section
   * @param data the data contained in the CData Section, may not be null
   * @throws XMLStreamException
   */
  public void writeCData(String data)
    throws XMLStreamException;

  /**
   * Write a DTD section.  This string represents the entire doctypedecl production
   * from the XML 1.0 specification.
   *
   * @param dtd the DTD to be written
   * @throws XMLStreamException
   */
  public void writeDTD(String dtd)
    throws XMLStreamException;

  /**
   * Writes an entity reference
   * @param name the name of the entity
   * @throws XMLStreamException
   */
  public void writeEntityRef(String name)
    throws XMLStreamException;

  /**
   * Write the XML Declaration. Defaults the XML version to 1.0, and the encoding to utf-8
   * @throws XMLStreamException
   */
  public void writeStartDocument()
    throws XMLStreamException;

  /**
   * Write the XML Declaration. Defaults the XML version to 1.0
   * @param version version of the xml document
   * @throws XMLStreamException
   */
  public void writeStartDocument(String version)
    throws XMLStreamException;

  /**
```

```
 * Write the XML Declaration.  Note that the encoding parameter does
 * not set the actual encoding of the underlying output.  That must
 * be set when the instance of the XMLStreamWriter is created using the
 * XMLOutputFactory
 * @param encoding encoding of the xml declaration
 * @param version version of the xml document
 * @throws XMLStreamException If given encoding does not match encoding
 * of the underlying stream
 */
public void writeStartDocument(String encoding,
                               String version)
  throws XMLStreamException;

/**
 * Write text to the output
 * @param text the value to write
 * @throws XMLStreamException
 */
public void writeCharacters(String text)
  throws XMLStreamException;

/**
 * Write text to the output
 * @param text the value to write
 * @param start the starting position in the array
 * @param len the number of characters to write
 * @throws XMLStreamException
 */
public void writeCharacters(char[] text, int start, int len)
  throws XMLStreamException;

/**
 * Gets the prefix the uri is bound to
 * @return the prefix or null
 * @throws XMLStreamException
 */
public String getPrefix(String uri)
  throws XMLStreamException;

/**
 * Sets the prefix the uri is bound to.  This prefix is bound
 * in the scope of the current START_ELEMENT / END_ELEMENT pair.
 * If this method is called before a START_ELEMENT has been written
 * the prefix is bound in the root scope.
 * @param prefix the prefix to bind to the uri, may not be null
 * @param uri the uri to bind to the prefix, may be null
 * @throws XMLStreamException
 */
public void setPrefix(String prefix, String uri)
  throws XMLStreamException;


/**
```

```
   * Binds a URI to the default namespace
   * This URI is bound
   * in the scope of the current START_ELEMENT / END_ELEMENT pair.
   * If this method is called before a START_ELEMENT has been written
   * the uri is bound in the root scope.
   * @param uri the uri to bind to the default namespace, may be null
   * @throws XMLStreamException
   */
  public void setDefaultNamespace(String uri)
    throws XMLStreamException;

  /**
   * Sets the current namespace context for prefix and uri bindings.
   * This context becomes the root namespace context for writing and
   * will replace the current root namespace context.  Subsequent calls
   * to setPrefix and setDefaultNamespace will bind namespaces using
   * the context passed to the method as the root context for resolving
   * namespaces.  This method may only be called once at the start of
   * the document.  It does not cause the namespaces to be declared.
   * If a namespace URI to prefix mapping is found in the namespace
   * context it is treated as declared and the prefix may be used
   * by the StreamWriter.
   * @param context the namespace context to use for this writer, may not be null
   * @throws XMLStreamException
   */
  public void setNamespaceContext(NamespaceContext context)
    throws XMLStreamException;

  /**
   * Returns the current namespace context.
   * @return the current NamespaceContext
   */
  public NamespaceContext getNamespaceContext();

  /**
   * Get the value of a feature/property from the underlying implementation
   * @param name The name of the property, may not be null
   * @return The value of the property
   * @throws IllegalArgumentException if the property is not supported
   * @throws NullPointerException if the name is null
   */
  public Object getProperty(java.lang.String name) throws IllegalArgumentException;

}
```

19 Xml Stream Events (javax.xml.stream.events package)

19.1 Attribute.java

Secuencia 207: javax/xml/stream/events/Attribute.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import javax.xml.namespace.QName;

/**
 * An interface that contains information about an attribute.  Attributes are reported
 * as a set of events accessible from a StartElement.  Other applications may report
 * Attributes as first-order events, for example as the results of an XPath expression.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see StartElement
 * @since 1.6
 */
public interface Attribute extends XMLEvent {

  /**
   * Returns the QName for this attribute
   */
  QName getName();

  /**
   * Gets the normalized value of this attribute
   */
  public String getValue();

  /**
   * Gets the type of this attribute, default is
```

```
   * the String "CDATA"
   * @return the type as a String, default is "CDATA"
   */
  public String getDTDType();

  /**
   * A flag indicating whether this attribute was actually
   * specified in the start-tag of its element, or was defaulted from the schema.
   * @return returns true if this was specified in the start element
   */
  public boolean isSpecified();

}
```

19.2 Characters.java

Secuencia 208: javax/xml/stream/events/Characters.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

/**
 * This describes the interface to Characters events.
 * All text events get reported as Characters events.
 * Content, CData and whitespace are all reported as
 * Characters events.  IgnorableWhitespace, in most cases,
 * will be set to false unless an element declaration of element
```

```
 * content is present for the current element.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface Characters extends XMLEvent {
  /**
   * Get the character data of this event
   */
  public String getData();

  /**
   * Returns true if this set of Characters
   * is all whitespace.  Whitespace inside a document
   * is reported as CHARACTERS.  This method allows
   * checking of CHARACTERS events to see if they
   * are composed of only whitespace characters
   */
  public boolean isWhiteSpace();

  /**
   * Returns true if this is a CData section.  If this
   * event is CData its event type will be CDATA
   *
   * If javax.xml.stream.isCoalescing is set to true CDATA Sections
   * that are surrounded by non CDATA characters will be reported
   * as a single Characters event. This method will return false
   * in this case.
   */
  public boolean isCData();

  /**
   * Return true if this is ignorableWhiteSpace.  If
   * this event is ignorableWhiteSpace its event type will
   * be SPACE.
   */
  public boolean isIgnorableWhiteSpace();

}
```

19.3 Comment.java

Secuencia 209: javax/xml/stream/events/Comment.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

/**
 * An interface for comment events
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface Comment extends XMLEvent {

  /**
   * Return the string data of the comment, returns empty string if it
   * does not exist
   */
  public String getText();
}
```

19.4 DTD.java

Secuencia 210: javax/xml/stream/events/DTD.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import java.util.List;

/**
 * This is the top level interface for events dealing with DTDs
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface DTD extends XMLEvent {

  /**
   * Returns the entire Document Type Declaration as a string, including
   * the internal DTD subset.
   * This may be null if there is not an internal subset.
   * If it is not null it must return the entire
   * Document Type Declaration which matches the doctypedecl
   * production in the XML 1.0 specification
   */
  String getDocumentTypeDeclaration();

  /**
   * Returns an implementation defined representation of the DTD.
   * This method may return null if no representation is available.
   */
  Object getProcessedDTD();

  /**
   * Return a List containing the notations declared in the DTD.
   * This list must contain NotationDeclaration events.
   * @see NotationDeclaration
   * @return an unordered list of NotationDeclaration events
   */
  List getNotations();

  /**
```

```
   * Return a List containing the general entities,
   * both external and internal, declared in the DTD.
   * This list must contain EntityDeclaration events.
   * @see EntityDeclaration
   * @return an unordered list of EntityDeclaration events
   */
  List getEntities();
}
```

19.5 EndDocument.java

Secuencia 211: javax/xml/stream/events/EndDocument.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

/**
 * A marker interface for the end of the document
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface EndDocument extends XMLEvent {
  /**
   * No methods are defined in this interface.
   */
```

```
}
```

19.6 EndElement.java

Secuencia 212: javax/xml/stream/events/EndElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import java.util.Iterator;
import javax.xml.namespace.QName;
/**
 * An interface for the end element event.  An EndElement is reported
 * for each End Tag in the document.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see XMLEvent
 * @since 1.6
 */
public interface EndElement extends XMLEvent {

  /**
   * Get the name of this event
   * @return the qualified name of this event
   */
  public QName getName();
```

```

  /**
   * Returns an Iterator of namespaces that have gone out
   * of scope.  Returns an empty iterator if no namespaces have gone
   * out of scope.
   * @return an Iterator over Namespace interfaces, or an
   * empty iterator
   */
  public Iterator getNamespaces();

}
```

19.7 EntityDeclaration.java

Secuencia 213: javax/xml/stream/events/EntityDeclaration.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;
/**
 * An interface for handling Entity Declarations
 *
 * This interface is used to record and report unparsed entity declarations.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
```

```
public interface EntityDeclaration extends XMLEvent {

  /**
   * The entity's public identifier, or null if none was given
   * @return the public ID for this declaration or null
   */
  String getPublicId();

  /**
   * The entity's system identifier.
   * @return the system ID for this declaration or null
   */
  String getSystemId();

  /**
   * The entity's name
   * @return the name, may not be null
   */
  String getName();

  /**
   * The name of the associated notation.
   * @return the notation name
   */
  String getNotationName();

  /**
   * The replacement text of the entity.
   * This method will only return non-null
   * if this is an internal entity.
   * @return null or the replacment text
   */
  String getReplacementText();

  /**
   * Get the base URI for this reference
   * or null if this information is not available
   * @return the base URI or null
   */
  String getBaseURI();

}
```

19.8 EntityReference.java

Secuencia 214: javax/xml/stream/events/EntityReference.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;
/**
 * An interface for handling Entity events.
 *
 * This event reports entities that have not been resolved
 * and reports their replacement text unprocessed (if
 * available).  This event will be reported if javax.xml.stream.isReplacingEntityReferences
 * is set to false.  If javax.xml.stream.isReplacingEntityReferences is set to true
 * entity references will be resolved transparently.
 *
 * Entities are handled in two possible ways:
 *
 * (1) If javax.xml.stream.isReplacingEntityReferences is set to true
 * all entity references are resolved and reported as markup transparently.
 * (2) If javax.xml.stream.isReplacingEntityReferences is set to false
 * Entity references are reported as an EntityReference Event.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface EntityReference extends XMLEvent {

  /**
   * Return the declaration of this entity.
   */
  EntityDeclaration getDeclaration();

  /**
   * The name of the entity
   * @return the entity's name, may not be null
   */
```

```
  String getName();
}
```

19.9 Namespace.java

Secuencia 215: javax/xml/stream/events/Namespace.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import javax.xml.namespace.QName;

/**
 * An interface that contains information about a namespace.
 * Namespaces are accessed from a StartElement.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see StartElement
 * @since 1.6
 */
public interface Namespace extends Attribute {

  /**
   * Gets the prefix, returns "" if this is a default
   * namespace declaration.
   */
```

```
  public String getPrefix();

  /**
   * Gets the uri bound to the prefix of this namespace
   */
  public String getNamespaceURI();

  /**
   * returns true if this attribute declares the default namespace
   */
  public boolean isDefaultNamespaceDeclaration();
}
```

19.10 NotationDeclaration.java

Secuencia 216: javax/xml/stream/events/NotationDeclaration.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;
/**
 * An interface for handling Notation Declarations
 *
 * Receive notification of a notation declaration event.
 * It is up to the application to record the notation for later reference,
 * At least one of publicId and systemId must be non-null.
 * There is no guarantee that the notation declaration
 * will be reported before any unparsed entities that use it.
```

```
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface NotationDeclaration extends XMLEvent {
  /**
   * The notation name.
   */
  String getName();

  /**
   * The notation's public identifier, or null if none was given.
   */
  String getPublicId();

  /**
   * The notation's system identifier, or null if none was given.
   */
  String getSystemId();
}
```

19.11 ProcessingInstruction.java

Secuencia 217: javax/xml/stream/events/ProcessingInstruction.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

```

```
package javax.xml.stream.events;
/**
 * An interface that describes the data found in processing instructions
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface ProcessingInstruction extends XMLEvent {

  /**
   * The target section of the processing instruction
   *
   * @return the String value of the PI or null
   */
  public String getTarget();

  /**
   * The data section of the processing instruction
   *
   * @return the String value of the PI's data or null
   */
  public String getData();
}
```

19.12 StartDocument.java

Secuencia 218: javax/xml/stream/events/StartDocument.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
```

```
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;
/**
 * An interface for the start document event
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface StartDocument extends XMLEvent {

  /**
   * Returns the system ID of the XML data
   * @return the system ID, defaults to ""
   */
  public String getSystemId();

  /**
   * Returns the encoding style of the XML data
   * @return the character encoding, defaults to "UTF-8"
   */
  public String getCharacterEncodingScheme();

  /**
   * Returns true if CharacterEncodingScheme was set in
   * the encoding declaration of the document
   */
  public boolean encodingSet();

  /**
   * Returns if this XML is standalone
   * @return the standalone state of XML, defaults to "no"
   */
  public boolean isStandalone();

  /**
   * Returns true if the standalone attribute was set in
   * the encoding declaration of the document.
   */
  public boolean standaloneSet();

  /**
   * Returns the version of XML of this XML stream
   * @return the version of XML, defaults to "1.0"
   */
  public String getVersion();
}
```

19.13 StartElement.java

Secuencia 219: javax/xml/stream/events/StartElement.java

```java
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import javax.xml.namespace.QName;
import javax.xml.namespace.NamespaceContext;

import java.util.Map;
import java.util.Iterator;

/**
 * The StartElement interface provides access to information about
 * start elements.  A StartElement is reported for each Start Tag
 * in the document.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface StartElement extends XMLEvent {

  /**
   * Get the name of this event
   * @return the qualified name of this event
   */
  public QName getName();

```

```
 /**
  * Returns an Iterator of non-namespace declared attributes declared on
  * this START_ELEMENT,
  * returns an empty iterator if there are no attributes.  The
  * iterator must contain only implementations of the javax.xml.stream.Attribute
  * interface.   Attributes are fundamentally unordered and may not be reported
  * in any order.
  *
  * @return a readonly Iterator over Attribute interfaces, or an
  * empty iterator
  */
 public Iterator getAttributes();

 /**
  * Returns an Iterator of namespaces declared on this element.
  * This Iterator does not contain previously declared namespaces
  * unless they appear on the current START_ELEMENT.
  * Therefore this list may contain redeclared namespaces and duplicate namespace
  * declarations. Use the getNamespaceContext() method to get the
  * current context of namespace declarations.
  *
  * <p>The iterator must contain only implementations of the
  * javax.xml.stream.Namespace interface.
  *
  * <p>A Namespace isA Attribute.  One
  * can iterate over a list of namespaces as a list of attributes.
  * However this method returns only the list of namespaces
  * declared on this START_ELEMENT and does not
  * include the attributes declared on this START_ELEMENT.
  *
  * Returns an empty iterator if there are no namespaces.
  *
  * @return a readonly Iterator over Namespace interfaces, or an
  * empty iterator
  *
  */
 public Iterator getNamespaces();

 /**
  * Returns the attribute referred to by this name
  * @param name the qname of the desired name
  * @return the attribute corresponding to the name value or null
  */
 public Attribute getAttributeByName(QName name);

 /**
  * Gets a read-only namespace context. If no context is
  * available this method will return an empty namespace context.
  * The NamespaceContext contains information about all namespaces
  * in scope for this StartElement.
  *
  * @return the current namespace context
  */
```

```
  public NamespaceContext getNamespaceContext();

  /**
   * Gets the value that the prefix is bound to in the
   * context of this element.  Returns null if
   * the prefix is not bound in this context
   * @param prefix the prefix to lookup
   * @return the uri bound to the prefix or null
   */
  public String getNamespaceURI(String prefix);
}
```

19.14 XMLEvent.java

Secuencia 220: javax/xml/stream/events/XMLEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.events;

import java.io.Writer;
import javax.xml.namespace.QName;
/**
 * This is the base event interface for handling markup events.
 * Events are value objects that are used to communicate the
 * XML 1.0 InfoSet to the Application.  Events may be cached
 * and referenced after the parse has completed.
 *
```

```
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.XMLEventReader
 * @see Characters
 * @see ProcessingInstruction
 * @see StartElement
 * @see EndElement
 * @see StartDocument
 * @see EndDocument
 * @see EntityReference
 * @see EntityDeclaration
 * @see NotationDeclaration
 * @since 1.6
 */
public interface XMLEvent extends javax.xml.stream.XMLStreamConstants {

  /**
   * Returns an integer code for this event.
   * @see #START_ELEMENT
   * @see #END_ELEMENT
   * @see #CHARACTERS
   * @see #ATTRIBUTE
   * @see #NAMESPACE
   * @see #PROCESSING_INSTRUCTION
   * @see #COMMENT
   * @see #START_DOCUMENT
   * @see #END_DOCUMENT
   * @see #DTD
   */
  public int getEventType();

  /**
   * Return the location of this event.  The Location
   * returned from this method is non-volatile and
   * will retain its information.
   * @see javax.xml.stream.Location
   */
  javax.xml.stream.Location getLocation();

  /**
   * A utility function to check if this event is a StartElement.
   * @see StartElement
   */
  public boolean isStartElement();

  /**
   * A utility function to check if this event is an Attribute.
   * @see Attribute
   */
  public boolean isAttribute();

  /**
   * A utility function to check if this event is a Namespace.
```

```
 * @see Namespace
 */
public boolean isNamespace();


/**
 * A utility function to check if this event is a EndElement.
 * @see EndElement
 */
public boolean isEndElement();

/**
 * A utility function to check if this event is an EntityReference.
 * @see EntityReference
 */
public boolean isEntityReference();

/**
 * A utility function to check if this event is a ProcessingInstruction.
 * @see ProcessingInstruction
 */
public boolean isProcessingInstruction();

/**
 * A utility function to check if this event is Characters.
 * @see Characters
 */
public boolean isCharacters();

/**
 * A utility function to check if this event is a StartDocument.
 * @see StartDocument
 */
public boolean isStartDocument();

/**
 * A utility function to check if this event is an EndDocument.
 * @see EndDocument
 */
public boolean isEndDocument();

/**
 * Returns this event as a start element event, may result in
 * a class cast exception if this event is not a start element.
 */
public StartElement asStartElement();

/**
 * Returns this event as an end  element event, may result in
 * a class cast exception if this event is not a end element.
 */
public EndElement asEndElement();

```

```
  /**
   * Returns this event as Characters, may result in
   * a class cast exception if this event is not Characters.
   */
  public Characters asCharacters();

  /**
   * This method is provided for implementations to provide
   * optional type information about the associated event.
   * It is optional and will return null if no information
   * is available.
   */
  public QName getSchemaType();

  /**
   * This method will write the XMLEvent as per the XML 1.0 specification as Unicode characters.
   * No indentation or whitespace should be outputted.
   *
   * Any user defined event type SHALL have this method
   * called when being written to on an output stream.
   * Built in Event types MUST implement this method,
   * but implementations MAY choose not call these methods
   * for optimizations reasons when writing out built in
   * Events to an output stream.
   * The output generated MUST be equivalent in terms of the
   * infoset expressed.
   *
   * @param writer The writer that will output the data
   * @throws XMLStreamException if there is a fatal error writing the event
   */
  public void writeAsEncodedUnicode(Writer writer)
    throws javax.xml.stream.XMLStreamException;

}
```

20 Xml Stream Util (javax.xml.stream.util package)

20.1 EventReaderDelegate.java

Secuencia 221: javax/xml/stream/util/EventReaderDelegate.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.util;

import javax.xml.namespace.QName;
import javax.xml.namespace.NamespaceContext;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.events.XMLEvent;
import javax.xml.stream.Location;
import javax.xml.stream.XMLStreamException;

/**
 * This is the base class for deriving an XMLEventReader
 * filter.
 *
 * This class is designed to sit between an XMLEventReader and an
 * application's XMLEventReader.  By default each method
 * does nothing but call the corresponding method on the
 * parent interface.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.XMLEventReader
 * @see StreamReaderDelegate
 * @since 1.6
 */

public class EventReaderDelegate implements XMLEventReader {
  private XMLEventReader reader;

  /**
   * Construct an empty filter with no parent.
   */
  public EventReaderDelegate(){}

  /**
   * Construct an filter with the specified parent.
   * @param reader the parent
   */
  public EventReaderDelegate(XMLEventReader reader) {
```

```
    this.reader = reader;
  }

  /**
   * Set the parent of this instance.
   * @param reader the new parent
   */
  public void setParent(XMLEventReader reader) {
    this.reader = reader;
  }

  /**
   * Get the parent of this instance.
   * @return the parent or null if none is set
   */
  public XMLEventReader getParent() {
    return reader;
  }

  public XMLEvent nextEvent()
    throws XMLStreamException
  {
    return reader.nextEvent();
  }

  public Object next() {
    return reader.next();
  }

  public boolean hasNext()
  {
    return reader.hasNext();
  }

  public XMLEvent peek()
    throws XMLStreamException
  {
    return reader.peek();
  }

  public void close()
    throws XMLStreamException
  {
    reader.close();
  }

  public String getElementText()
    throws XMLStreamException
  {
    return reader.getElementText();
  }

  public XMLEvent nextTag()
```

```
    throws XMLStreamException
  {
    return reader.nextTag();
  }

  public Object getProperty(java.lang.String name)
    throws java.lang.IllegalArgumentException
  {
    return reader.getProperty(name);
  }

  public void remove() {
    reader.remove();
  }
}
```

20.2 StreamReaderDelegate.java

Secuencia 222: javax/xml/stream/util/StreamReaderDelegate.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.util;

import java.io.Reader;
import javax.xml.namespace.QName;
import javax.xml.namespace.NamespaceContext;
import javax.xml.stream.XMLStreamReader;
```

```java
import javax.xml.stream.Location;
import javax.xml.stream.XMLStreamException;

/**
 * This is the base class for deriving an XMLStreamReader filter
 *
 * This class is designed to sit between an XMLStreamReader and an
 * application's XMLStreamReader.   By default each method
 * does nothing but call the corresponding method on the
 * parent interface.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.XMLStreamReader
 * @see EventReaderDelegate
 * @since 1.6
 */

public class StreamReaderDelegate implements XMLStreamReader {
  private XMLStreamReader reader;

  /**
   * Construct an empty filter with no parent.
   */
  public StreamReaderDelegate(){}

  /**
   * Construct an filter with the specified parent.
   * @param reader the parent
   */
  public StreamReaderDelegate(XMLStreamReader reader) {
    this.reader = reader;
  }

  /**
   * Set the parent of this instance.
   * @param reader the new parent
   */
  public void setParent(XMLStreamReader reader) {
    this.reader = reader;
  }

  /**
   * Get the parent of this instance.
   * @return the parent or null if none is set
   */
  public XMLStreamReader getParent() {
    return reader;
  }

  public int next()
    throws XMLStreamException
  {
```

```
    return reader.next();
  }

  public int nextTag()
    throws XMLStreamException
  {
    return reader.nextTag();
  }

  public String getElementText()
    throws XMLStreamException
  {
    return reader.getElementText();
  }

  public void require(int type, String namespaceURI, String localName)
    throws XMLStreamException
  {
    reader.require(type,namespaceURI,localName);
  }

  public boolean hasNext()
    throws XMLStreamException
  {
    return reader.hasNext();
  }

  public void close()
    throws XMLStreamException
  {
    reader.close();
  }

  public String getNamespaceURI(String prefix)
  {
    return reader.getNamespaceURI(prefix);
  }

  public NamespaceContext getNamespaceContext() {
    return reader.getNamespaceContext();
  }

  public boolean isStartElement() {
    return reader.isStartElement();
  }

  public boolean isEndElement() {
    return reader.isEndElement();
  }

  public boolean isCharacters() {
    return reader.isCharacters();
  }
```

```java

  public boolean isWhiteSpace() {
    return reader.isWhiteSpace();
  }

  public String getAttributeValue(String namespaceUri,
                                  String localName)
  {
    return reader.getAttributeValue(namespaceUri,localName);
  }

  public int getAttributeCount() {
    return reader.getAttributeCount();
  }

  public QName getAttributeName(int index) {
    return reader.getAttributeName(index);
  }

  public String getAttributePrefix(int index) {
    return reader.getAttributePrefix(index);
  }

  public String getAttributeNamespace(int index) {
    return reader.getAttributeNamespace(index);
  }

  public String getAttributeLocalName(int index) {
    return reader.getAttributeLocalName(index);
  }

  public String getAttributeType(int index) {
    return reader.getAttributeType(index);
  }

  public String getAttributeValue(int index) {
    return reader.getAttributeValue(index);
  }

  public boolean isAttributeSpecified(int index) {
    return reader.isAttributeSpecified(index);
  }

  public int getNamespaceCount() {
    return reader.getNamespaceCount();
  }

  public String getNamespacePrefix(int index) {
    return reader.getNamespacePrefix(index);
  }

  public String getNamespaceURI(int index) {
    return reader.getNamespaceURI(index);
```

```
  }

  public int getEventType() {
    return reader.getEventType();
  }

  public String getText() {
    return reader.getText();
  }

  public int getTextCharacters(int sourceStart,
                               char[] target,
                               int targetStart,
                               int length)
    throws XMLStreamException {
    return reader.getTextCharacters(sourceStart,
                                    target,
                                    targetStart,
                                    length);
  }


  public char[] getTextCharacters() {
    return reader.getTextCharacters();
  }

  public int getTextStart() {
    return reader.getTextStart();
  }

  public int getTextLength() {
    return reader.getTextLength();
  }

  public String getEncoding() {
    return reader.getEncoding();
  }

  public boolean hasText() {
    return reader.hasText();
  }

  public Location getLocation() {
    return reader.getLocation();
  }

  public QName getName() {
    return reader.getName();
  }

  public String getLocalName() {
    return reader.getLocalName();
  }
```

```

  public boolean hasName() {
    return reader.hasName();
  }

  public String getNamespaceURI() {
    return reader.getNamespaceURI();
  }

  public String getPrefix() {
    return reader.getPrefix();
  }

  public String getVersion() {
    return reader.getVersion();
  }

  public boolean isStandalone() {
    return reader.isStandalone();
  }

  public boolean standaloneSet() {
    return reader.standaloneSet();
  }

  public String getCharacterEncodingScheme() {
    return reader.getCharacterEncodingScheme();
  }

  public String getPITarget() {
    return reader.getPITarget();
  }

  public String getPIData() {
    return reader.getPIData();
  }

  public Object getProperty(String name) {
    return reader.getProperty(name);
  }
}
```

20.3 XMLEventAllocator.java

Secuencia 223: javax/xml/stream/util/XMLEventAllocator.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

package javax.xml.stream.util;

import javax.xml.stream.events.XMLEvent;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.XMLStreamException;

/**
 * This interface defines a class that allows a user to register
 * a way to allocate events given an XMLStreamReader.  An implementation
 * is not required to use the XMLEventFactory implementation but this
 * is recommended.  The XMLEventAllocator can be set on an XMLInputFactory
 * using the property "javax.xml.stream.allocator"
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @see javax.xml.stream.XMLInputFactory
 * @see javax.xml.stream.XMLEventFactory
 * @since 1.6
 */
public interface XMLEventAllocator {

  /**
   * This method creates an instance of the XMLEventAllocator. This
   * allows the XMLInputFactory to allocate a new instance per reader.
   */
  public XMLEventAllocator newInstance();

  /**
   * This method allocates an event given the current
   * state of the XMLStreamReader.  If this XMLEventAllocator
   * does not have a one-to-one mapping between reader states
   * and events this method will return null.  This method
   * must not modify the state of the XMLStreamReader.
```

```
   * @param reader The XMLStreamReader to allocate from
   * @return the event corresponding to the current reader state
   */
  public XMLEvent allocate(XMLStreamReader reader)
    throws XMLStreamException;

  /**
   * This method allocates an event or set of events
   * given the current
   * state of the XMLStreamReader and adds the event
   * or set of events to the
   * consumer that was passed in.  This method can be used
   * to expand or contract reader states into event states.
   * This method may modify the state of the XMLStreamReader.
   * @param reader The XMLStreamReader to allocate from
   * @param consumer The XMLEventConsumer to add to.
   */
  public void allocate(XMLStreamReader reader, XMLEventConsumer consumer)
    throws XMLStreamException;

}
```

20.4 XMLEventConsumer.java

Secuencia 224: javax/xml/stream/util/XMLEventConsumer.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 * Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 */

```

```java
package javax.xml.stream.util;

import javax.xml.stream.events.XMLEvent;
import javax.xml.stream.XMLStreamException;

/**
 * This interface defines an event consumer interface.  The contract of the
 * of a consumer is to accept the event.  This interface can be used to
 * mark an object as able to receive events.  Add may be called several
 * times in immediate succession so a consumer must be able to cache
 * events it hasn't processed yet.
 *
 * @version 1.0
 * @author Copyright (c) 2009 by Oracle Corporation. All Rights Reserved.
 * @since 1.6
 */
public interface XMLEventConsumer {

  /**
   * This method adds an event to the consumer. Calling this method
   * invalidates the event parameter. The client application should
   * discard all references to this event upon calling add.
   * The behavior of an application that continues to use such references
   * is undefined.
   *
   * @param event the event to add, may not be null
   */
  public void add(XMLEvent event)
    throws XMLStreamException;
}
```

21 Xml Transform (javax.xml.transform package)

21.1 ErrorListener.java

Secuencia 225: javax/xml/transform/ErrorListener.java

```java
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
 * <p>To provide customized error handling, implement this interface and
 * use the <code>setErrorListener</code> method to register an instance of the
 * implmentation with the {@link javax.xml.transform.Transformer}.  The
 * <code>Transformer</code> then reports all errors and warnings through this
 * interface.</p>
 *
 * <p>If an application does <em>not</em> register its own custom
 * <code>ErrorListener</code>, the default <code>ErrorListener</code>
 * is used which reports all warnings and errors to <code>System.err</code>
 * and does not throw any <code>Exception</code>s.
 * Applications are <em>strongly</em> encouraged to register and use
 * <code>ErrorListener</code>s that insure proper behavior for warnings and
 * errors.</p>
 *
 * <p>For transformation errors, a <code>Transformer</code> must use this
 * interface instead of throwing an <code>Exception</code>: it is up to the
 * application to decide whether to throw an <code>Exception</code> for
 * different types of errors and warnings.  Note however that the
 * <code>Transformer</code> is not required to continue with the transformation
 * after a call to {@link #fatalError(TransformerException exception)}.</p>
 *
 * <p><code>Transformer</code>s may use this mechanism to report XML parsing
 * errors as well as transformation errors.</p>
 */
public interface ErrorListener {

    /**
     * Receive notification of a warning.
     *
     * <p>{@link javax.xml.transform.Transformer} can use this method to report
     * conditions that are not errors or fatal errors.  The default behaviour
     * is to take no action.</p>
     *
     * <p>After invoking this method, the Transformer must continue with
     * the transformation. It should still be possible for the
     * application to process the document through to the end.</p>
     *
     * @param exception The warning information encapsulated in a
     *                  transformer exception.
     *
     * @throws javax.xml.transform.TransformerException if the application
     * chooses to discontinue the transformation.
```

```
     *
     * @see javax.xml.transform.TransformerException
     */
    public abstract void warning(TransformerException exception)
        throws TransformerException;

    /**
     * Receive notification of a recoverable error.
     *
     * <p>The transformer must continue to try and provide normal transformation
     * after invoking this method.  It should still be possible for the
     * application to process the document through to the end if no other errors
     * are encountered.</p>
     *
     * @param exception The error information encapsulated in a
     *                  transformer exception.
     *
     * @throws javax.xml.transform.TransformerException if the application
     * chooses to discontinue the transformation.
     *
     * @see javax.xml.transform.TransformerException
     */
    public abstract void error(TransformerException exception)
        throws TransformerException;

    /**
     * <p>Receive notification of a non-recoverable error.</p>
     *
     * <p>The processor may choose to continue, but will not normally
     * proceed to a successful completion.</p>
     *
     * <p>The method should throw an exception if it is unable to
     * process the error, or if it wishes execution to terminate
     * immediately. The processor will not necessarily honor this
     * request.</p>
     *
     * @param exception The error information encapsulated in a
     *    <code>TransformerException</code>.
     *
     * @throws javax.xml.transform.TransformerException if the application
     * chooses to discontinue the transformation.
     *
     * @see javax.xml.transform.TransformerException
     */
    public abstract void fatalError(TransformerException exception)
        throws TransformerException;
}
```

21.2 FactoryFinder.java

Secuencia 226: javax/xml/transform/FactoryFinder.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

import java.io.File;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Iterator;
import java.util.Properties;
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * <p>Implements pluggable Datatypes.</p>
 *
 * <p>This class is duplicated for each JAXP subpackage so keep it in
 * sync.  It is package private for secure class loading.</p>
 *
 * @author Santiago.PericasGeertsen@sun.com
 * @author Huizhe.Wang@oracle.com
 */
class FactoryFinder {
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xalan.internal.";

    /**
     * Internal debug flag.
     */
    private static boolean debug = false;

    /**
```

```
     * Cache for properties in java.home/lib/jaxp.properties
     */
    private final static Properties cacheProps = new Properties();

    /**
     * Flag indicating if properties from java.home/lib/jaxp.properties
     * have been cached.
     */
    static volatile boolean firstTime = true;

    /**
     * Security support class use to check access control before
     * getting certain system resources.
     */
    private final static SecuritySupport ss = new SecuritySupport();

    // Define system property "jaxp.debug" to get output
    static {
        // Use try/catch block to support applets, which throws
        // SecurityException out of this code.
        try {
            String val = ss.getSystemProperty("jaxp.debug");
            // Allow simply setting the prop to turn on debug
            debug = val != null && !"false".equals(val);
        }
        catch (SecurityException se) {
            debug = false;
        }
    }

    private static void dPrint(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * Attempt to load a class using the class loader supplied. If that fails
     * and fall back is enabled, the current (i.e. bootstrap) class loader is
     * tried.
     *
     * If the class loader supplied is <code>null</code>, first try using the
     * context class loader followed by the current (i.e. bootstrap) class
     * loader.
     *
     * Use bootstrap classLoader if cl = null and useBSClsLoader is true
     */
    static private Class<?> getProviderClass(String className, ClassLoader cl,
            boolean doFallback, boolean useBSClsLoader) throws ClassNotFoundException
    {
        try {
            if (cl == null) {
                if (useBSClsLoader) {
```

```
                    return Class.forName(className, false, FactoryFinder.class.getClassLoader());
                } else {
                    cl = ss.getContextClassLoader();
                    if (cl == null) {
                        throw new ClassNotFoundException();
                    }
                    else {
                        return Class.forName(className, false, cl);
                    }
                }
            }
            else {
                return Class.forName(className, false, cl);
            }
        }
        catch (ClassNotFoundException e1) {
            if (doFallback) {
                // Use current class loader - should always be bootstrap CL
                return Class.forName(className, false, FactoryFinder.class.getClassLoader());
            }
            else {
                throw e1;
            }
        }
    }

    /**
     * Create an instance of a class. Delegates to method
     * <code>getProviderClass()</code> in order to load the class.
     *
     * @param type Base class / Service interface  of the factory to
     *             instantiate.
     *
     * @param className Name of the concrete class corresponding to the
     * service provider
     *
     * @param cl <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     * current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     * @param doFallback True if the current ClassLoader should be tried as
     * a fallback if the class is not found using cl
     *
     * @param useServicesMechanism True use services mechanism
     */
    static <T> T newInstance(Class<T> type, String className, ClassLoader cl,
                             boolean doFallback, boolean useServicesMechanism)
        throws TransformerFactoryConfigurationError
    {
        assert type != null;

        boolean useBSClsLoader = false;
        // make sure we have access to restricted packages
        if (System.getSecurityManager() != null) {
```

```
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                cl = null;
                useBSClsLoader = true;
            }
        }

        try {
            Class<?> providerClass = getProviderClass(className, cl, doFallback, useBSClsLoader);
            if (!type.isAssignableFrom(providerClass)) {
                throw new ClassCastException(className + " cannot be cast to " + type.getName());
            }
            Object instance = null;
            if (!useServicesMechanism) {
                instance = newInstanceNoServiceLoader(type, providerClass);
            }
            if (instance == null) {
                instance = providerClass.newInstance();
            }
            if (debug) {    // Extra check to avoid computing cl strings
                dPrint("created new instance of " + providerClass +
                       " using ClassLoader: " + cl);
            }
            return type.cast(instance);
        }
        catch (ClassNotFoundException x) {
            throw new TransformerFactoryConfigurationError(x,
                "Provider " + className + " not found");
        }
        catch (Exception x) {
            throw new TransformerFactoryConfigurationError(x,
                "Provider " + className + " could not be instantiated: " + x);
        }
    }

    /**
     * Try to construct using newTransformerFactoryNoServiceLoader
     *   method if available.
     */
    private static <T> T newInstanceNoServiceLoader(Class<T> type, Class<?> providerClass) {
        // Retain maximum compatibility if no security manager.
        if (System.getSecurityManager() == null) {
            return null;
        }
        try {
            final Method creationMethod =
                    providerClass.getDeclaredMethod(
                        "newTransformerFactoryNoServiceLoader"
                    );
            final int modifiers = creationMethod.getModifiers();

            // Do not call the method if it's not public static.
            if (!Modifier.isPublic(modifiers) || !Modifier.isStatic(modifiers)) {
                return null;
```

```
            }

            // Only call the method if it's declared to return an instance of
            // TransformerFactory
            final Class<?> returnType = creationMethod.getReturnType();
            if (type.isAssignableFrom(returnType)) {
                final Object result = creationMethod.invoke(null, (Object[])null);
                return type.cast(result);
            } else {
                // This should not happen, as
                // TransformerFactoryImpl.newTransformerFactoryNoServiceLoader is
                // declared to return TransformerFactory.
                throw new ClassCastException(returnType + " cannot be cast to " + type);
            }
        } catch (ClassCastException e) {
            throw new TransformerFactoryConfigurationError(e, e.getMessage());
        } catch (NoSuchMethodException exc) {
            return null;
        } catch (Exception exc) {
            return null;
        }
    }

    /**
     * Finds the implementation Class object in the specified order.  Main
     * entry point.
     * @return Class object of factory, never null
     *
     * @param type                  Base class / Service interface  of the
     *                              factory to find.
     *
     * @param fallbackClassName     Implementation class name, if nothing else
     *                              is found.  Use null to mean no fallback.
     *
     * Package private so this code can be shared.
     */
    static <T> T find(Class<T> type, String fallbackClassName)
        throws TransformerFactoryConfigurationError
    {
        assert type != null;

        final String factoryId = type.getName();

        dPrint("find factoryId =" + factoryId);
        // Use the system property first
        try {
            String systemProp = ss.getSystemProperty(factoryId);
            if (systemProp != null) {
                dPrint("found system property, value=" + systemProp);
                return newInstance(type, systemProp, null, true, true);
            }
        }
        catch (SecurityException se) {
```

```
            if (debug) se.printStackTrace();
        }

        // try to read from $java.home/lib/jaxp.properties
        try {
            if (firstTime) {
                synchronized (cacheProps) {
                    if (firstTime) {
                        String configFile = ss.getSystemProperty("java.home") + File.separator +
                            "lib" + File.separator + "jaxp.properties";
                        File f = new File(configFile);
                        firstTime = false;
                        if (ss.doesFileExist(f)) {
                            dPrint("Read properties file "+f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                    }
                }
            }
            final String factoryClassName = cacheProps.getProperty(factoryId);

            if (factoryClassName != null) {
                dPrint("found in $java.home/jaxp.properties, value=" + factoryClassName);
                return newInstance(type, factoryClassName, null, true, true);
            }
        }
        catch (Exception ex) {
            if (debug) ex.printStackTrace();
        }

        // Try Jar Service Provider Mechanism
        T provider = findServiceProvider(type);
        if (provider != null) {
            return provider;
        }
        if (fallbackClassName == null) {
            throw new TransformerFactoryConfigurationError(null,
                "Provider for " + factoryId + " cannot be found");
        }

        dPrint("loaded from fallback value: " + fallbackClassName);
        return newInstance(type, fallbackClassName, null, true, true);
    }

    /*
     * Try to find provider using the ServiceLoader.
     *
     * @param type Base class / Service interface  of the factory to find.
     *
     * @return instance of provider class if found or null
     */
    private static <T> T findServiceProvider(final Class<T> type)
        throws TransformerFactoryConfigurationError
```

```
    {
      try {
            return AccessController.doPrivileged(new PrivilegedAction<T>() {
                public T run() {
                    final ServiceLoader<T> serviceLoader = ServiceLoader.load(type);
                    final Iterator<T> iterator = serviceLoader.iterator();
                    if (iterator.hasNext()) {
                        return iterator.next();
                    } else {
                        return null;
                    }
                }
            });
        } catch(ServiceConfigurationError e) {
            // It is not possible to wrap an error directly in
            // FactoryConfigurationError - so we need to wrap the
            // ServiceConfigurationError in a RuntimeException.
            // The alternative would be to modify the logic in
            // FactoryConfigurationError to allow setting a
            // Throwable as the cause, but that could cause
            // compatibility issues down the road.
            final RuntimeException x = new RuntimeException(
                    "Provider for " + type + " cannot be created", e);
            final TransformerFactoryConfigurationError error =
                    new TransformerFactoryConfigurationError(x, x.getMessage());
            throw error;
        }
    }
}
```

21.3 OutputKeys.java

Secuencia 227: javax/xml/transform/OutputKeys.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

package javax.xml.transform;

/**
 * Provides string constants that can be used to set
 * output properties for a Transformer, or to retrieve
 * output properties from a Transformer or Templates object.
 * <p>All the fields in this class are read-only.</p>
 *
 * @see <a href="http://www.w3.org/TR/xslt#output">
 *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
 */
public class OutputKeys {

    /**
     * Default constructor is private on purpose.  This class is
     * only for static variable access, and should never be constructed.
     */
    private OutputKeys() { }

    /**
     * method = "xml" | "html" | "text" | <var>expanded name</var>.
     *
     * <p>The value of the method property identifies the overall method that
     * should be used for outputting the result tree.  Other non-namespaced
     * values may be used, such as "xhtml", but, if accepted, the handling
     * of such values is implementation defined.  If any of the method values
     * are not accepted and are not namespace qualified,
     * then {@link javax.xml.transform.Transformer#setOutputProperty}
     * or {@link javax.xml.transform.Transformer#setOutputProperties} will
     * throw a {@link java.lang.IllegalArgumentException}.</p>
     *
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String METHOD = "method";

    /**
     * version = <var>nmtoken</var>.
     *
     * <p><code>version</code> specifies the version of the output
     * method.</p>
     * <p>When the output method is "xml", the version value specifies the
     * version of XML to be used for outputting the result tree. The default
     * value for the xml output method is 1.0. When the output method is
     * "html", the version value indicates the version of the HTML.
     * The default value for the xml output method is 4.0, which specifies
     * that the result should be output as HTML conforming to the HTML 4.0
     * Recommendation [HTML].  If the output method is "text", the version
```

```
     * property is ignored.</p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String VERSION = "version";

    /**
     * encoding = <var>string</var>.
     *
     * <p><code>encoding</code> specifies the preferred character
     * encoding that the Transformer should use to encode sequences of
     * characters as sequences of bytes. The value of the encoding property should be
     * treated case-insensitively. The value must only contain characters in
     * the range #x21 to #x7E (i.e., printable ASCII characters). The value
     * should either be a <code>charset</code> registered with the Internet
     * Assigned Numbers Authority <a href="http://www.iana.org/">[IANA]</a>,
     * <a href="http://www.ietf.org/rfc/rfc2278.txt">[RFC2278]</a>
     * or start with <code>X-</code>.</p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     * section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String ENCODING = "encoding";

    /**
     * omit-xml-declaration = "yes" | "no".
     *
     * <p><code>omit-xml-declaration</code> specifies whether the XSLT
     * processor should output an XML declaration; the value must be
     * <code>yes</code> or <code>no</code>.</p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String OMIT_XML_DECLARATION = "omit-xml-declaration";

    /**
     * standalone = "yes" | "no".
     *
     * <p><code>standalone</code> specifies whether the Transformer
     * should output a standalone document declaration; the value must be
     * <code>yes</code> or <code>no</code>.</p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String STANDALONE = "standalone";

    /**
     * doctype-public = <var>string</var>.
     * <p>See the documentation for the {@link #DOCTYPE_SYSTEM} property
     * for a description of what the value of the key should be.</p>
     *
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
```

```
    public static final String DOCTYPE_PUBLIC = "doctype-public";

    /**
     * doctype-system = <var>string</var>.
     * <p><code>doctype-system</code> specifies the system identifier
     * to be used in the document type declaration.</p>
     * <p>If the doctype-system property is specified, the xml output method
     * should output a document type declaration immediately before the first
     * element. The name following &lt;!DOCTYPE should be the name of the first
     * element. If doctype-public property is also specified, then the xml
     * output method should output PUBLIC followed by the public identifier
     * and then the system identifier; otherwise, it should output SYSTEM
     * followed by the system identifier. The internal subset should be empty.
     * The value of the doctype-public property should be ignored unless the doctype-system
     * property is specified.</p>
     * <p>If the doctype-public or doctype-system properties are specified,
     * then the html output method should output a document type declaration
     * immediately before the first element. The name following &lt;!DOCTYPE
     * should be HTML or html. If the doctype-public property is specified,
     * then the output method should output PUBLIC followed by the specified
     * public identifier; if the doctype-system property is also specified,
     * it should also output the specified system identifier following the
     * public identifier. If the doctype-system property is specified but
     * the doctype-public property is not specified, then the output method
     * should output SYSTEM followed by the specified system identifier.</p>
     *
     * <p><code>doctype-system</code> specifies the system identifier
     * to be used in the document type declaration.</p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String DOCTYPE_SYSTEM = "doctype-system";

    /**
     * cdata-section-elements = <var>expanded names</var>.
     *
     * <p><code>cdata-section-elements</code> specifies a whitespace delimited
     * list of the names of elements whose text node children should be output
     * using CDATA sections. Note that these names must use the format
     * described in the section Qualfied Name Representation in
     * {@link javax.xml.transform}.</p>
     *
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation.</a>
     */
    public static final String CDATA_SECTION_ELEMENTS =
        "cdata-section-elements";

    /**
     * indent = "yes" | "no".
     *
     * <p><code>indent</code> specifies whether the Transformer may
     * add additional whitespace when outputting the result tree; the value
```

```
     * must be <code>yes</code> or <code>no</code>.  </p>
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *  section 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String INDENT = "indent";

    /**
     * media-type = <var>string</var>.
     *
     * <p><code>media-type</code> specifies the media type (MIME
     * content type) of the data that results from outputting the result
     * tree. The <code>charset</code> parameter should not be specified
     * explicitly; instead, when the top-level media type is
     * <code>text</code>, a <code>charset</code> parameter should be added
     * according to the character encoding actually used by the output
     * method.  </p>
     * @see <a href="http://www.w3.org/TR/xslt#output">s
     * ection 16 of the XSL Transformations (XSLT) W3C Recommendation</a>
     */
    public static final String MEDIA_TYPE = "media-type";
}
```

21.4 Result.java

Secuencia 228: javax/xml/transform/Result.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
```

```
 * <p>An object that implements this interface contains the information
 * needed to build a transformation result tree.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public interface Result {

    /**
     * The name of the processing instruction that is sent if the
     * result tree disables output escaping.
     *
     * <p>Normally, result tree serialization escapes & and < (and
     * possibly other characters) when outputting text nodes.
     * This ensures that the output is well-formed XML. However,
     * it is sometimes convenient to be able to produce output that is
     * almost, but not quite well-formed XML; for example,
     * the output may include ill-formed sections that will
     * be transformed into well-formed XML by a subsequent non-XML aware
     * process. If a processing instruction is sent with this name,
     * serialization should be output without any escaping. </p>
     *
     * <p>Result DOM trees may also have PI_DISABLE_OUTPUT_ESCAPING and
     * PI_ENABLE_OUTPUT_ESCAPING inserted into the tree.</p>
     *
     * @see <a href="http://www.w3.org/TR/xslt#disable-output-escaping">disable-output-escaping in
         XSLT Specification</a>
     */
    public static final String PI_DISABLE_OUTPUT_ESCAPING =
        "javax.xml.transform.disable-output-escaping";

    /**
     * The name of the processing instruction that is sent
     * if the result tree enables output escaping at some point after having
     * received a PI_DISABLE_OUTPUT_ESCAPING processing instruction.
     *
     * @see <a href="http://www.w3.org/TR/xslt#disable-output-escaping">disable-output-escaping in
         XSLT Specification</a>
     */
    public static final String PI_ENABLE_OUTPUT_ESCAPING =
        "javax.xml.transform.enable-output-escaping";

    /**
     * Set the system identifier for this Result.
     *
     * <p>If the Result is not to be written to a file, the system identifier is optional.
     * The application may still want to provide one, however, for use in error messages
     * and warnings, or to resolve relative output identifiers.</p>
     *
     * @param systemId The system identifier as a URI string.
     */
    public void setSystemId(String systemId);

    /**
```

```
     * Get the system identifier that was set with setSystemId.
     *
     * @return The system identifier that was set with setSystemId,
     * or null if setSystemId was not called.
     */
    public String getSystemId();
}
```

21.5 SecuritySupport.java

Secuencia 229: javax/xml/transform/SecuritySupport.java

```
/*
 * Copyright (c) 2004, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

import java.security.*;
import java.net.*;
import java.io.*;
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


```

```
    ClassLoader getContextClassLoader() throws SecurityException{
        return (ClassLoader)
        AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                //try {
                cl = Thread.currentThread().getContextClassLoader();
                //} catch (SecurityException ex) { }
                if (cl == null)
                    cl = ClassLoader.getSystemClassLoader();
                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                    final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
```

```
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

}
```

21.6 Source.java

Secuencia 230: javax/xml/transform/Source.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
 * An object that implements this interface contains the information
 * needed to act as source input (XML source or transformation instructions).
 */
public interface Source {

    /**
     * Set the system identifier for this Source.
     *
```

```
     * <p>The system identifier is optional if the source does not
     * get its data from a URL, but it may still be useful to provide one.
     * The application can use a system identifier, for example, to resolve
     * relative URIs and to include in error messages and warnings.</p>
     *
     * @param systemId The system identifier as a URL string.
     */
    public void setSystemId(String systemId);

    /**
     * Get the system identifier that was set with setSystemId.
     *
     * @return The system identifier that was set with setSystemId, or null
     * if setSystemId was not called.
     */
    public String getSystemId();
}
```

21.7 SourceLocator.java

Secuencia 231: javax/xml/transform/SourceLocator.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
 * This interface is primarily for the purposes of reporting where
 * an error occurred in the XML source or transformation instructions.
 */
public interface SourceLocator {
```

```

    /**
     * Return the public identifier for the current document event.
     *
     * <p>The return value is the public identifier of the document
     * entity or of the external parsed entity in which the markup that
     * triggered the event appears.</p>
     *
     * @return A string containing the public identifier, or
     *         null if none is available.
     * @see #getSystemId
     */
    public String getPublicId();

    /**
     * Return the system identifier for the current document event.
     *
     * <p>The return value is the system identifier of the document
     * entity or of the external parsed entity in which the markup that
     * triggered the event appears.</p>
     *
     * <p>If the system identifier is a URL, the parser must resolve it
     * fully before passing it to the application.</p>
     *
     * @return A string containing the system identifier, or null
     *         if none is available.
     * @see #getPublicId
     */
    public String getSystemId();

    /**
     * Return the line number where the current document event ends.
     *
     * <p><strong>Warning:</strong> The return value from the method
     * is intended only as an approximation for the sake of error
     * reporting; it is not intended to provide sufficient information
     * to edit the character content of the original XML document.</p>
     *
     * <p>The return value is an approximation of the line number
     * in the document entity or external parsed entity where the
     * markup that triggered the event appears.</p>
     *
     * @return The line number, or -1 if none is available.
     * @see #getColumnNumber
     */
    public int getLineNumber();

    /**
     * Return the character position where the current document event ends.
     *
     * <p><strong>Warning:</strong> The return value from the method
     * is intended only as an approximation for the sake of error
     * reporting; it is not intended to provide sufficient information
```

```
     * to edit the character content of the original XML document.</p>
     *
     * <p>The return value is an approximation of the column number
     * in the document entity or external parsed entity where the
     * markup that triggered the event appears.</p>
     *
     * @return The column number, or -1 if none is available.
     * @see #getLineNumber
     */
    public int getColumnNumber();
}
```

21.8 Templates.java

Secuencia 232: javax/xml/transform/Templates.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

import java.util.Properties;




/**
 * An object that implements this interface is the runtime representation of processed
 * transformation instructions.
 *
 * <p>Templates must be threadsafe for a given instance
 * over multiple threads running concurrently, and may
```

```
 * be used multiple times in a given session.</p>
 */
public interface Templates {

    /**
     * Create a new transformation context for this Templates object.
     *
     * @return A valid non-null instance of a Transformer.
     *
     * @throws TransformerConfigurationException if a Transformer can not be created.
     */
    Transformer newTransformer() throws TransformerConfigurationException;

    /**
     * Get the properties corresponding to the effective xsl:output element.
     * The object returned will
     * be a clone of the internal values. Accordingly, it can be mutated
     * without mutating the Templates object, and then handed in to
     * {@link javax.xml.transform.Transformer#setOutputProperties}.
     *
     * <p>The properties returned should contain properties set by the stylesheet,
     * and these properties are "defaulted" by default properties specified by
     * <a href="http://www.w3.org/TR/xslt#output">section 16 of the
     * XSL Transformations (XSLT) W3C Recommendation</a>.  The properties that
     * were specifically set by the stylesheet should be in the base
     * Properties list, while the XSLT default properties that were not
     * specifically set should be in the "default" Properties list.  Thus,
     * getOutputProperties().getProperty(String key) will obtain any
     * property in that was set by the stylesheet, <em>or</em> the default
     * properties, while
     * getOutputProperties().get(String key) will only retrieve properties
     * that were explicitly set in the stylesheet.</p>
     *
     * <p>For XSLT,
     * <a href="http://www.w3.org/TR/xslt#attribute-value-templates">Attribute
     * Value Templates</a> attribute values will
     * be returned unexpanded (since there is no context at this point).  The
     * namespace prefixes inside Attribute Value Templates will be unexpanded,
     * so that they remain valid XPath values.</p>
     *
     * @return A Properties object, never null.
     */
    Properties getOutputProperties();
}
```

21.9 Transformer.java

Secuencia 233: javax/xml/transform/Transformer.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

import java.util.Properties;

/**
 * An instance of this abstract class can transform a
 * source tree into a result tree.
 *
 * <p>An instance of this class can be obtained with the
 * {@link TransformerFactory#newTransformer TransformerFactory.newTransformer}
 * method. This instance may then be used to process XML from a
 * variety of sources and write the transformation output to a
 * variety of sinks.</p>
 *
 * <p>An object of this class may not be used in multiple threads
 * running concurrently.  Different Transformers may be used
 * concurrently by different threads.</p>
 *
 * <p>A <code>Transformer</code> may be used multiple times.  Parameters and
 * output properties are preserved across transformations.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public abstract class Transformer {

    /**
     * Default constructor is protected on purpose.
     */
    protected Transformer() { }

        /**
         * <p>Reset this <code>Transformer</code> to its original configuration.</p>
         *
```

```
     * <p><code>Transformer</code> is reset to the same state as when it was created with
     * {@link TransformerFactory#newTransformer()},
     * {@link TransformerFactory#newTransformer(Source source)} or
     * {@link Templates#newTransformer()}.
     * <code>reset()</code> is designed to allow the reuse of existing <code>Transformer</code>s
     * thus saving resources associated with the creation of new <code>Transformer</code>s.</p>
     *
     * <p>The reset <code>Transformer</code> is not guaranteed to have the same {@link URIResolver}
     * or {@link ErrorListener} <code>Object</code>s, e.g. {@link Object#equals(Object obj)}.
     * It is guaranteed to have a functionally equal <code>URIResolver</code>
     * and <code>ErrorListener</code>.</p>
     *
     * @throws UnsupportedOperationException When implementation does not
     *   override this method.
     *
     * @since 1.5
     */
    public void reset() {

            // implementors should override this method
            throw new UnsupportedOperationException(
                    "This Transformer, \"" + this.getClass().getName() + "\", does not support the reset functionality."
                    + "  Specification \"" + this.getClass().getPackage().getSpecificationTitle() + "\""
                    + " version \"" + this.getClass().getPackage().getSpecificationVersion() + "\""
                    );
    }

    /**
     * <p>Transform the XML <code>Source</code> to a <code>Result</code>.
     * Specific transformation behavior is determined by the settings of the
     * <code>TransformerFactory</code> in effect when the
     * <code>Transformer</code> was instantiated and any modifications made to
     * the <code>Transformer</code> instance.</p>
     *
     * <p>An empty <code>Source</code> is represented as an empty document
     * as constructed by {@link javax.xml.parsers.DocumentBuilder#newDocument()}.
     * The result of transforming an empty <code>Source</code> depends on
     * the transformation behavior; it is not always an empty
     * <code>Result</code>.</p>
     *
     * @param xmlSource The XML input to transform.
     * @param outputTarget The <code>Result</code> of transforming the
     *   <code>xmlSource</code>.
     *
     * @throws TransformerException If an unrecoverable error occurs
     *   during the course of the transformation.
     */
    public abstract void transform(Source xmlSource, Result outputTarget)
```

```
        throws TransformerException;

    /**
     * Add a parameter for the transformation.
     *
     * <p>Pass a qualified name as a two-part string, the namespace URI
     * enclosed in curly braces ({}), followed by the local name. If the
     * name has a null URL, the String only contain the local name. An
     * application can safely check for a non-null URI by testing to see if the
     * first character of the name is a '{' character.</p>
     * <p>For example, if a URI and local name were obtained from an element
     * defined with &lt;xyz:foo
     * xmlns:xyz="http://xyz.foo.com/yada/baz.html"/&gt;,
     * then the qualified name would be "{http://xyz.foo.com/yada/baz.html}foo".
     * Note that no prefix is used.</p>
     *
     * @param name The name of the parameter, which may begin with a
     * namespace URI in curly braces ({}).
     * @param value The value object.  This can be any valid Java object. It is
     * up to the processor to provide the proper object coersion or to simply
     * pass the object on for use in an extension.
     *
     * @throws NullPointerException If value is null.
     */
     public abstract void setParameter(String name, Object value);

    /**
     * Get a parameter that was explicitly set with setParameter.
     *
     * <p>This method does not return a default parameter value, which
     * cannot be determined until the node context is evaluated during
     * the transformation process.
     *
     * @param name of <code>Object</code> to get
     *
     * @return A parameter that has been set with setParameter.
     */
    public abstract Object getParameter(String name);

    /**
     * <p>Set a list of parameters.</p>
     *
     * <p>Note that the list of parameters is specified as a
     * <code>Properties</code> <code>Object</code> which limits the parameter
     * values to <code>String</code>s.  Multiple calls to
     * {@link #setParameter(String name, Object value)} should be used when the
     * desired values are non-<code>String</code> <code>Object</code>s.
     * The parameter names should conform as specified in
     * {@link #setParameter(String name, Object value)}.
     * An <code>IllegalArgumentException</code> is thrown if any names do not
     * conform.</p>
     *
     * <p>New parameters in the list are added to any existing parameters.
```

```
     * If the name of a new parameter is equal to the name of an existing
     * parameter as determined by {@link java.lang.Object#equals(Object obj)},
     *  the existing parameter is set to the new value.</p>
     *
     * @param params Parameters to set.
     *
     * @throws IllegalArgumentException If any parameter names do not conform
     *   to the naming rules.
     */

    /**
     * Clear all parameters set with setParameter.
     */
    public abstract void clearParameters();

    /**
     * Set an object that will be used to resolve URIs used in
     * document().
     *
     * <p>If the resolver argument is null, the URIResolver value will
     * be cleared and the transformer will no longer have a resolver.</p>
     *
     * @param resolver An object that implements the URIResolver interface,
     * or null.
     */
    public abstract void setURIResolver(URIResolver resolver);

    /**
     * Get an object that will be used to resolve URIs used in
     * document().
     *
     * @return An object that implements the URIResolver interface,
     * or null.
     */
    public abstract URIResolver getURIResolver();

    /**
     * Set the output properties for the transformation.  These
     * properties will override properties set in the Templates
     * with xsl:output.
     *
     * <p>If argument to this function is null, any properties
     * previously set are removed, and the value will revert to the value
     * defined in the templates object.</p>
     *
     * <p>Pass a qualified property key name as a two-part string, the namespace
     * URI enclosed in curly braces ({}), followed by the local name. If the
     * name has a null URL, the String only contain the local name. An
     * application can safely check for a non-null URI by testing to see if the
     * first character of the name is a '{' character.</p>
     * <p>For example, if a URI and local name were obtained from an element
     * defined with &lt;xyz:foo
     * xmlns:xyz="http://xyz.foo.com/yada/baz.html"/&gt;,
```

```
     * then the qualified name would be "{http://xyz.foo.com/yada/baz.html}foo".
     * Note that no prefix is used.</p>
     * An <code>IllegalArgumentException</code> is thrown  if any of the
     * argument keys are not recognized and are not namespace qualified.
     *
     * @param oformat A set of output properties that will be
     *   used to override any of the same properties in affect
     *   for the transformation.
     *
     * @throws IllegalArgumentException When keys are not recognized and
     *   are not namespace qualified.
     *
     * @see javax.xml.transform.OutputKeys
     * @see java.util.Properties
     *
     */
    public abstract void setOutputProperties(Properties oformat);

    /**
     * <p>Get a copy of the output properties for the transformation.</p>
     *
     * <p>The properties returned should contain properties set by the user,
     * and properties set by the stylesheet, and these properties
     * are "defaulted" by default properties specified by
     * <a href="http://www.w3.org/TR/xslt#output">section 16 of the
     * XSL Transformations (XSLT) W3C Recommendation</a>.  The properties that
     * were specifically set by the user or the stylesheet should be in the base
     * Properties list, while the XSLT default properties that were not
     * specifically set should be the default Properties list.  Thus,
     * getOutputProperties().getProperty(String key) will obtain any
     * property in that was set by {@link #setOutputProperty},
     * {@link #setOutputProperties}, in the stylesheet, <em>or</em> the default
     * properties, while
     * getOutputProperties().get(String key) will only retrieve properties
     * that were explicitly set by {@link #setOutputProperty},
     * {@link #setOutputProperties}, or in the stylesheet.</p>
     *
     * <p>Note that mutation of the Properties object returned will not
     * effect the properties that the transformer contains.</p>
     *
     * <p>If any of the argument keys are not recognized and are not
     * namespace qualified, the property will be ignored and not returned.
     * In other words the behaviour is not orthogonal with
     * {@link #setOutputProperties setOutputProperties}.</p>
     *
     * @return A copy of the set of output properties in effect for
     *   the next transformation.
     *
     * @see javax.xml.transform.OutputKeys
     * @see java.util.Properties
     * @see <a href="http://www.w3.org/TR/xslt#output">
     *   XSL Transformations (XSLT) Version 1.0</a>
     */
```

```
    public abstract Properties getOutputProperties();

    /**
     * Set an output property that will be in effect for the
     * transformation.
     *
     * <p>Pass a qualified property name as a two-part string, the namespace URI
     * enclosed in curly braces ({}), followed by the local name. If the
     * name has a null URL, the String only contain the local name. An
     * application can safely check for a non-null URI by testing to see if the
     * first character of the name is a '{' character.</p>
     * <p>For example, if a URI and local name were obtained from an element
     * defined with &lt;xyz:foo
     * xmlns:xyz="http://xyz.foo.com/yada/baz.html"/&gt;,
     * then the qualified name would be "{http://xyz.foo.com/yada/baz.html}foo".
     * Note that no prefix is used.</p>
     *
     * <p>The Properties object that was passed to {@link #setOutputProperties}
     * won't be effected by calling this method.</p>
     *
     * @param name A non-null String that specifies an output
     * property name, which may be namespace qualified.
     * @param value The non-null string value of the output property.
     *
     * @throws IllegalArgumentException If the property is not supported, and is
     * not qualified with a namespace.
     *
     * @see javax.xml.transform.OutputKeys
     */
    public abstract void setOutputProperty(String name, String value)
        throws IllegalArgumentException;

    /**
     * <p>Get an output property that is in effect for the transformer.</p>
     *
     * <p>If a property has been set using {@link #setOutputProperty},
     * that value will be returned. Otherwise, if a property is explicitly
     * specified in the stylesheet, that value will be returned. If
     * the value of the property has been defaulted, that is, if no
     * value has been set explicitly either with {@link #setOutputProperty} or
     * in the stylesheet, the result may vary depending on
     * implementation and input stylesheet.</p>
     *
     * @param name A non-null String that specifies an output
     * property name, which may be namespace qualified.
     *
     * @return The string value of the output property, or null
     * if no property was found.
     *
     * @throws IllegalArgumentException If the property is not supported.
     *
     * @see javax.xml.transform.OutputKeys
     */
```

```
    public abstract String getOutputProperty(String name)
        throws IllegalArgumentException;

    /**
     * Set the error event listener in effect for the transformation.
     *
     * @param listener The new error listener.
     *
     * @throws IllegalArgumentException if listener is null.
     */
    public abstract void setErrorListener(ErrorListener listener)
        throws IllegalArgumentException;

    /**
     * Get the error event handler in effect for the transformation.
     * Implementations must provide a default error listener.
     *
     * @return The current error handler, which should never be null.
     */
    public abstract ErrorListener getErrorListener();
}
```

21.10 TransformerConfigurationException.java

Secuencia 234: javax/xml/transform/TransformerConfigurationException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
```

```
 * Indicates a serious configuration error.
 */
public class TransformerConfigurationException extends TransformerException {

    /**
     * Create a new <code>TransformerConfigurationException</code> with no
     * detail mesage.
     */
    public TransformerConfigurationException() {
        super("Configuration Error");
    }

    /**
     * Create a new <code>TransformerConfigurationException</code> with
     * the <code>String </code> specified as an error message.
     *
     * @param msg The error message for the exception.
     */
    public TransformerConfigurationException(String msg) {
        super(msg);
    }

    /**
     * Create a new <code>TransformerConfigurationException</code> with a
     * given <code>Exception</code> base cause of the error.
     *
     * @param e The exception to be encapsulated in a
     * TransformerConfigurationException.
     */
    public TransformerConfigurationException(Throwable e) {
        super(e);
    }

    /**
     * Create a new <code>TransformerConfigurationException</code> with the
     * given <code>Exception</code> base cause and detail message.
     *
     * @param e The exception to be encapsulated in a
     *      TransformerConfigurationException
     * @param msg The detail message.
     */
    public TransformerConfigurationException(String msg, Throwable e) {
        super(msg, e);
    }

    /**
     * Create a new TransformerConfigurationException from a message and a Locator.
     *
     * <p>This constructor is especially useful when an application is
     * creating its own exception from within a DocumentHandler
     * callback.</p>
     *
     * @param message The error or warning message.
```

```
     * @param locator The locator object for the error or warning.
     */
    public TransformerConfigurationException(String message,
                                             SourceLocator locator) {
        super(message, locator);
    }

    /**
     * Wrap an existing exception in a TransformerConfigurationException.
     *
     * @param message The error or warning message, or null to
     *                use the message from the embedded exception.
     * @param locator The locator object for the error or warning.
     * @param e Any exception.
     */
    public TransformerConfigurationException(String message,
                                             SourceLocator locator,
                                             Throwable e) {
        super(message, locator, e);
    }
}
```

21.11 TransformerException.java

Secuencia 235: javax/xml/transform/TransformerException.java

```
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

import java.lang.reflect.Method;
```

```
import java.lang.reflect.InvocationTargetException;

/**
 * This class specifies an exceptional condition that occured
 * during the transformation process.
 */
public class TransformerException extends Exception {

    /** Field locator specifies where the error occured */
    SourceLocator locator;

    /**
     * Method getLocator retrieves an instance of a SourceLocator
     * object that specifies where an error occured.
     *
     * @return A SourceLocator object, or null if none was specified.
     */
    public SourceLocator getLocator() {
        return locator;
    }

    /**
     * Method setLocator sets an instance of a SourceLocator
     * object that specifies where an error occured.
     *
     * @param location A SourceLocator object, or null to clear the location.
     */
    public void setLocator(SourceLocator location) {
        locator = location;
    }

    /** Field containedException specifies a wrapped exception.  May be null. */
    Throwable containedException;

    /**
     * This method retrieves an exception that this exception wraps.
     *
     * @return An Throwable object, or null.
     * @see #getCause
     */
    public Throwable getException() {
        return containedException;
    }

    /**
     * Returns the cause of this throwable or <code>null</code> if the
     * cause is nonexistent or unknown.  (The cause is the throwable that
     * caused this throwable to get thrown.)
     */
    public Throwable getCause() {

        return ((containedException == this)
                ? null
```

```
                : containedException);
    }

    /**
     * Initializes the <i>cause</i> of this throwable to the specified value.
     * (The cause is the throwable that caused this throwable to get thrown.)
     *
     * <p>This method can be called at most once.  It is generally called from
     * within the constructor, or immediately after creating the
     * throwable.  If this throwable was created
     * with {@link #TransformerException(Throwable)} or
     * {@link #TransformerException(String,Throwable)}, this method cannot be called
     * even once.
     *
     * @param  cause the cause (which is saved for later retrieval by the
     *         {@link #getCause()} method).  (A <code>null</code> value is
     *         permitted, and indicates that the cause is nonexistent or
     *         unknown.)
     * @return  a reference to this <code>Throwable</code> instance.
     * @throws IllegalArgumentException if <code>cause</code> is this
     *         throwable.  (A throwable cannot
     *         be its own cause.)
     * @throws IllegalStateException if this throwable was
     *         created with {@link #TransformerException(Throwable)} or
     *         {@link #TransformerException(String,Throwable)}, or this method has already
     *         been called on this throwable.
     */
    public synchronized Throwable initCause(Throwable cause) {

        if (this.containedException != null) {
            throw new IllegalStateException("Can't overwrite cause");
        }

        if (cause == this) {
            throw new IllegalArgumentException(
                "Self-causation not permitted");
        }

        this.containedException = cause;

        return this;
    }

    /**
     * Create a new TransformerException.
     *
     * @param message The error or warning message.
     */
    public TransformerException(String message) {

        super(message);

        this.containedException = null;
```

```
        this.locator            = null;
    }

    /**
     * Create a new TransformerException wrapping an existing exception.
     *
     * @param e The exception to be wrapped.
     */
    public TransformerException(Throwable e) {

        super(e.toString());

        this.containedException = e;
        this.locator            = null;
    }

    /**
     * Wrap an existing exception in a TransformerException.
     *
     * <p>This is used for throwing processor exceptions before
     * the processing has started.</p>
     *
     * @param message The error or warning message, or null to
     *                use the message from the embedded exception.
     * @param e Any exception
     */
    public TransformerException(String message, Throwable e) {

        super(((message == null) || (message.length() == 0))
              ? e.toString()
              : message);

        this.containedException = e;
        this.locator            = null;
    }

    /**
     * Create a new TransformerException from a message and a Locator.
     *
     * <p>This constructor is especially useful when an application is
     * creating its own exception from within a DocumentHandler
     * callback.</p>
     *
     * @param message The error or warning message.
     * @param locator The locator object for the error or warning.
     */
    public TransformerException(String message, SourceLocator locator) {

        super(message);

        this.containedException = null;
        this.locator            = locator;
    }
```

```

    /**
     * Wrap an existing exception in a TransformerException.
     *
     * @param message The error or warning message, or null to
     *                use the message from the embedded exception.
     * @param locator The locator object for the error or warning.
     * @param e Any exception
     */
    public TransformerException(String message, SourceLocator locator,
                                Throwable e) {

        super(message);

        this.containedException = e;
        this.locator            = locator;
    }

    /**
     * Get the error message with location information
     * appended.
     *
     * @return A <code>String</code> representing the error message with
     *         location information appended.
     */
    public String getMessageAndLocation() {

        StringBuffer sbuffer = new StringBuffer();
        String       message = super.getMessage();

        if (null != message) {
            sbuffer.append(message);
        }

        if (null != locator) {
            String systemID = locator.getSystemId();
            int    line     = locator.getLineNumber();
            int    column   = locator.getColumnNumber();

            if (null != systemID) {
                sbuffer.append("; SystemID: ");
                sbuffer.append(systemID);
            }

            if (0 != line) {
                sbuffer.append("; Line#: ");
                sbuffer.append(line);
            }

            if (0 != column) {
                sbuffer.append("; Column#: ");
                sbuffer.append(column);
            }
```

```
        }

        return sbuffer.toString();
    }

    /**
     * Get the location information as a string.
     *
     * @return A string with location info, or null
     * if there is no location information.
     */
    public String getLocationAsString() {

        if (null != locator) {
            StringBuffer sbuffer  = new StringBuffer();
            String       systemID = locator.getSystemId();
            int          line     = locator.getLineNumber();
            int          column   = locator.getColumnNumber();

            if (null != systemID) {
                sbuffer.append("; SystemID: ");
                sbuffer.append(systemID);
            }

            if (0 != line) {
                sbuffer.append("; Line#: ");
                sbuffer.append(line);
            }

            if (0 != column) {
                sbuffer.append("; Column#: ");
                sbuffer.append(column);
            }

            return sbuffer.toString();
        } else {
            return null;
        }
    }

    /**
     * Print the the trace of methods from where the error
     * originated.  This will trace all nested exception
     * objects, as well as this object.
     */
    public void printStackTrace() {
        printStackTrace(new java.io.PrintWriter(System.err, true));
    }

    /**
     * Print the the trace of methods from where the error
     * originated.  This will trace all nested exception
     * objects, as well as this object.
```

```
     * @param s The stream where the dump will be sent to.
     */
    public void printStackTrace(java.io.PrintStream s) {
        printStackTrace(new java.io.PrintWriter(s));
    }

    /**
     * Print the the trace of methods from where the error
     * originated.  This will trace all nested exception
     * objects, as well as this object.
     * @param s The writer where the dump will be sent to.
     */
    public void printStackTrace(java.io.PrintWriter s) {

        if (s == null) {
            s = new java.io.PrintWriter(System.err, true);
        }

        try {
            String locInfo = getLocationAsString();

            if (null != locInfo) {
                s.println(locInfo);
            }

            super.printStackTrace(s);
        } catch (Throwable e) {}

        Throwable exception = getException();

        for (int i = 0; (i < 10) && (null != exception); i++) {
            s.println("---------");

            try {
                if (exception instanceof TransformerException) {
                    String locInfo =
                        ((TransformerException) exception)
                            .getLocationAsString();

                    if (null != locInfo) {
                        s.println(locInfo);
                    }
                }

                exception.printStackTrace(s);
            } catch (Throwable e) {
                s.println("Could not print stack trace...");
            }

            try {
                Method meth =
                    ((Object) exception).getClass().getMethod("getException",
                        (Class[]) null);
```

```

                if (null != meth) {
                    Throwable prev = exception;

                    exception = (Throwable) meth.invoke(exception, (Object[]) null);

                    if (prev == exception) {
                        break;
                    }
                } else {
                    exception = null;
                }
            } catch (InvocationTargetException ite) {
                exception = null;
            } catch (IllegalAccessException iae) {
                exception = null;
            } catch (NoSuchMethodException nsme) {
                exception = null;
            }
        }
        // insure output is written
        s.flush();
    }
}
```

21.12 TransformerFactory.java

Secuencia 236: javax/xml/transform/TransformerFactory.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```java
package javax.xml.transform;

/**
 * <p>A TransformerFactory instance can be used to create
 * {@link javax.xml.transform.Transformer} and
 * {@link javax.xml.transform.Templates} objects.</p>
 *
 * <p>The system property that determines which Factory implementation
 * to create is named <code>"javax.xml.transform.TransformerFactory"</code>.
 * This property names a concrete subclass of the
 * <code>TransformerFactory</code> abstract class. If the property is not
 * defined, a platform default is be used.</p>
 *
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @author <a href="mailto:Neeraj.Bajaj@sun.com">Neeraj Bajaj</a>
 *
 * @since 1.5
 */
public abstract class TransformerFactory {

    /**
     * Default constructor is protected on purpose.
     */
    protected TransformerFactory() { }



    /**
     * <p>Obtain a new instance of a <code>TransformerFactory</code>.
     * This static method creates a new factory instance.</p>
     * <p>This method uses the following ordered lookup procedure to determine
     * the <code>TransformerFactory</code> implementation class to
     * load:</p>
     * <ul>
     * <li>
     * Use the <code>javax.xml.transform.TransformerFactory</code> system
     * property.
     * </li>
     * <li>
     * Use the properties file "lib/jaxp.properties" in the JRE directory.
     * This configuration file is in standard <code>java.util.Properties
     * </code> format and contains the fully qualified name of the
     * implementation class with the key being the system property defined
     * above.
     * <br>
     * The jaxp.properties file is read only once by the JAXP implementation
     * and it's values are then cached for future use.  If the file does not exist
     * when the first attempt is made to read from it, no further attempts are
     * made to check for its existence.  It is not possible to change the value
     * of any property in jaxp.properties after it has been read for the first time.
     * </li>
     * <li>
     *   Use the service-provider loading facilities, defined by the
```

```
     *   {@link java.util.ServiceLoader} class, to attempt to locate and load an
     *   implementation of the service using the {@linkplain
     *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
     *   the service-provider loading facility will use the {@linkplain
     *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
     *   to attempt to load the service. If the context class
     *   loader is null, the {@linkplain
     *   ClassLoader#getSystemClassLoader() system class loader} will be used.
     * </li>
     * <li>
     *   Otherwise, the system-default implementation is returned.
     * </li>
     * </ul>
     *
     * <p>Once an application has obtained a reference to a <code>
     * TransformerFactory</code> it can use the factory to configure
     * and obtain transformer instances.</p>
     *
     * @return new TransformerFactory instance, never null.
     *
     * @throws TransformerFactoryConfigurationError Thrown in case of {@linkplain
     * java.util.ServiceConfigurationError service configuration error} or if
     * the implementation is not available or cannot be instantiated.
     */
    public static TransformerFactory newInstance()
        throws TransformerFactoryConfigurationError {

        return FactoryFinder.find(
            /* The default property name according to the JAXP spec */
            TransformerFactory.class,
            /* The fallback implementation class name, XSLTC */
            "com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl");
    }

    /**
     * <p>Obtain a new instance of a <code>TransformerFactory</code> from factory class name.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
     * should be loaded.</p>
     *
     * <p>Once an application has obtained a reference to a <code>
     * TransformerFactory</code> it can use the factory to configure
     * and obtain transformer instances.</p>
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
```

```
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <
         code>javax.xml.transform.TransformerFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</
         code>
     *                     current <code>Thread</code>'s context classLoader is used to load the
         factory class.
     *
     * @return new TransformerFactory instance, never null.
     *
     * @throws TransformerFactoryConfigurationError
     *                    if <code>factoryClassName</code> is <code>null</code>, or
     *                   the factory class cannot be loaded, instantiated.
     *
     * @see #newInstance()
     *
     * @since 1.6
     */
    public static TransformerFactory newInstance(String factoryClassName, ClassLoader classLoader)
        throws TransformerFactoryConfigurationError{

        //do not fallback if given classloader can't find the class, throw exception
        return  FactoryFinder.newInstance(TransformerFactory.class,
                    factoryClassName, classLoader, false, false);
    }
    /**
     * <p>Process the <code>Source</code> into a <code>Transformer</code>
     * <code>Object</code>.  The <code>Source</code> is an XSLT document that
     * conforms to <a href="http://www.w3.org/TR/xslt">
     * XSL Transformations (XSLT) Version 1.0</a>.  Care must
     * be taken not to use this <code>Transformer</code> in multiple
     * <code>Thread</code>s running concurrently.
     * Different <code>TransformerFactories</code> can be used concurrently by
     * different <code>Thread</code>s.</p>
     *
     * @param source <code>Source </code> of XSLT document used to create
     *   <code>Transformer</code>.
     *   Examples of XML <code>Source</code>s include
     *   {@link javax.xml.transform.dom.DOMSource DOMSource},
     *   {@link javax.xml.transform.sax.SAXSource SAXSource}, and
     *   {@link javax.xml.transform.stream.StreamSource StreamSource}.
     *
     * @return A <code>Transformer</code> object that may be used to perform
     *   a transformation in a single <code>Thread</code>, never
     *   <code>null</code>.
     *
     * @throws TransformerConfigurationException Thrown if there are errors when
     *    parsing the <code>Source</code> or it is not possible to create a
     *   <code>Transformer</code> instance.
     *
     * @see <a href="http://www.w3.org/TR/xslt">
     *   XSL Transformations (XSLT) Version 1.0</a>
```

```
     */
    public abstract Transformer newTransformer(Source source)
        throws TransformerConfigurationException;

    /**
     * <p>Create a new <code>Transformer</code> that performs a copy
     * of the <code>Source</code> to the <code>Result</code>.
     * i.e. the "<em>identity transform</em>".</p>
     *
     * @return A Transformer object that may be used to perform a transformation
     * in a single thread, never null.
     *
     * @throws TransformerConfigurationException When it is not
     *   possible to create a <code>Transformer</code> instance.
     */
    public abstract Transformer newTransformer()
        throws TransformerConfigurationException;

    /**
     * Process the Source into a Templates object, which is a
     * a compiled representation of the source. This Templates object
     * may then be used concurrently across multiple threads.  Creating
     * a Templates object allows the TransformerFactory to do detailed
     * performance optimization of transformation instructions, without
     * penalizing runtime transformation.
     *
     * @param source An object that holds a URL, input stream, etc.
     *
     * @return A Templates object capable of being used for transformation
     *   purposes, never <code>null</code>.
     *
     * @throws TransformerConfigurationException When parsing to
     *   construct the Templates object fails.
     */
    public abstract Templates newTemplates(Source source)
        throws TransformerConfigurationException;

    /**
     * <p>Get the stylesheet specification(s) associated with the
     * XML <code>Source</code> document via the
     * <a href="http://www.w3.org/TR/xml-stylesheet/">
     * xml-stylesheet processing instruction</a> that match the given criteria.
     * Note that it is possible to return several stylesheets, in which case
     * they are applied as if they were a list of imports or cascades in a
     * single stylesheet.</p>
     *
     * @param source The XML source document.
     * @param media The media attribute to be matched.  May be null, in which
     *      case the prefered templates will be used (i.e. alternate = no).
     * @param title The value of the title attribute to match.  May be null.
     * @param charset The value of the charset attribute to match.  May be null.
     *
     * @return A <code>Source</code> <code>Object</code> suitable for passing
```

```
     *   to the <code>TransformerFactory</code>.
     *
     * @throws TransformerConfigurationException An <code>Exception</code>
     *   is thrown if an error occurings during parsing of the
     *   <code>source</code>.
     *
     * @see <a href="http://www.w3.org/TR/xml-stylesheet/">
     *   Associating Style Sheets with XML documents Version 1.0</a>
     */
    public abstract Source getAssociatedStylesheet(
        Source source,
        String media,
        String title,
        String charset)
        throws TransformerConfigurationException;

    /**
     * Set an object that is used by default during the transformation
     * to resolve URIs used in document(), xsl:import, or xsl:include.
     *
     * @param resolver An object that implements the URIResolver interface,
     * or null.
     */
    public abstract void setURIResolver(URIResolver resolver);

    /**
     * Get the object that is used by default during the transformation
     * to resolve URIs used in document(), xsl:import, or xsl:include.
     *
     * @return The URIResolver that was set with setURIResolver.
     */
    public abstract URIResolver getURIResolver();

    //======= CONFIGURATION METHODS =======

        /**
         * <p>Set a feature for this <code>TransformerFactory</code> and <code>Transformer</code>s
         * or <code>Template</code>s created by this factory.</p>
         *
         * <p>
         * Feature names are fully qualified {@link java.net.URI}s.
         * Implementations may define their own features.
         * An {@link TransformerConfigurationException} is thrown if this <code>TransformerFactory
             </code> or the
         * <code>Transformer</code>s or <code>Template</code>s it creates cannot support the
             feature.
         * It is possible for an <code>TransformerFactory</code> to expose a feature value but be
             unable to change its state.
         * </p>
         *
         * <p>All implementations are required to support the {@link javax.xml.XMLConstants#
             FEATURE_SECURE_PROCESSING} feature.
         * When the feature is:</p>
```

```
        * <ul>
        *   <li>
        *     <code>true</code>: the implementation will limit XML processing to conform to implementation limits
        *     and behave in a secure fashion as defined by the implementation.
        *     Examples include resolving user defined style sheets and functions.
        *     If XML processing is limited for security reasons, it will be reported via a call to the registered
        *     {@link ErrorListener#fatalError(TransformerException exception)}.
        *     See {@link  #setErrorListener(ErrorListener listener)}.
        *   </li>
        *   <li>
        *     <code>false</code>: the implementation will processing XML according to the XML specifications without
        *     regard to possible implementation limits.
        *   </li>
        * </ul>
        *
        * @param name Feature name.
        * @param value Is feature state <code>true</code> or <code>false</code>.
        *
        * @throws TransformerConfigurationException if this <code>TransformerFactory</code>
        *   or the <code>Transformer</code>s or <code>Template</code>s it creates cannot support this feature.
     * @throws NullPointerException If the <code>name</code> parameter is null.
        */
       public abstract void setFeature(String name, boolean value)
               throws TransformerConfigurationException;

    /**
     * Look up the value of a feature.
     *
        * <p>
        * Feature names are fully qualified {@link java.net.URI}s.
        * Implementations may define their own features.
        * <code>false</code> is returned if this <code>TransformerFactory</code> or the
        * <code>Transformer</code>s or <code>Template</code>s it creates cannot support the feature.
        * It is possible for an <code>TransformerFactory</code> to expose a feature value but be unable to change its state.
        * </p>
        *
        * @param name Feature name.
        *
     * @return The current state of the feature, <code>true</code> or <code>false</code>.
     *
     * @throws NullPointerException If the <code>name</code> parameter is null.
     */
    public abstract boolean getFeature(String name);

    /**
     * Allows the user to set specific attributes on the underlying
     * implementation.  An attribute in this context is defined to
```

```
     * be an option that the implementation provides.
     * An <code>IllegalArgumentException</code> is thrown if the underlying
     * implementation doesn't recognize the attribute.
     * <p>
     * All implementations that implement JAXP 1.5 or newer are required to
     * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD}  and
     * {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_STYLESHEET} properties.
     * </p>
     * <ul>
     *   <li>
     *      <p>
     *      Access to external DTDs in the source file is restricted to the protocols
     *      specified by the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property.
     *      If access is denied during transformation due to the restriction of this property,
     *      {@link javax.xml.transform.TransformerException} will be thrown by
     *      {@link javax.xml.transform.Transformer#transform(Source, Result)}.
     *      </p>
     *      <p>
     *      Access to external DTDs in the stylesheet is restricted to the protocols
     *      specified by the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property.
     *      If access is denied during the creation of a new transformer due to the
     *      restriction of this property,
     *      {@link javax.xml.transform.TransformerConfigurationException} will be thrown
     *      by the {@link #newTransformer(Source)} method.
     *      </p>
     *      <p>
     *      Access to external reference set by the stylesheet processing instruction,
     *      Import and Include element is restricted to the protocols specified by the
     *      {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_STYLESHEET} property.
     *      If access is denied during the creation of a new transformer due to the
     *      restriction of this property,
     *      {@link javax.xml.transform.TransformerConfigurationException} will be thrown
     *      by the {@link #newTransformer(Source)} method.
     *      </p>
     *      <p>
     *      Access to external document through XSLT document function is restricted
     *      to the protocols specified by the property. If access is denied during
     *      the transformation due to the restriction of this property,
     *      {@link javax.xml.transform.TransformerException} will be thrown by the
     *      {@link javax.xml.transform.Transformer#transform(Source, Result)} method.
     *      </p>
     *   </li>
     * </ul>
     *
     * @param name The name of the attribute.
     * @param value The value of the attribute.
     *
     * @throws IllegalArgumentException When implementation does not
     *   recognize the attribute.
     */
    public abstract void setAttribute(String name, Object value);

    /**
```

```
     * Allows the user to retrieve specific attributes on the underlying
     * implementation.
     * An <code>IllegalArgumentException</code> is thrown if the underlying
     * implementation doesn't recognize the attribute.
     *
     * @param name The name of the attribute.
     *
     * @return value The value of the attribute.
     *
     * @throws IllegalArgumentException When implementation does not
     *   recognize the attribute.
     */
    public abstract Object getAttribute(String name);

    /**
     * Set the error event listener for the TransformerFactory, which
     * is used for the processing of transformation instructions,
     * and not for the transformation itself.
     * An <code>IllegalArgumentException</code> is thrown if the
     * <code>ErrorListener</code> listener is <code>null</code>.
     *
     * @param listener The new error listener.
     *
     * @throws IllegalArgumentException When <code>listener</code> is
     *   <code>null</code>
     */
    public abstract void setErrorListener(ErrorListener listener);

    /**
     * Get the error event handler for the TransformerFactory.
     *
     * @return The current error handler, which should never be null.
     */
    public abstract ErrorListener getErrorListener();

}
```

21.13 TransformerFactoryConfigurationError.java

Secuencia 237: javax/xml/transform/TransformerFactoryConfigurationError.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
 * Thrown when a problem with configuration with the Transformer Factories
 * exists. This error will typically be thrown when the class of a
 * transformation factory specified in the system properties cannot be found
 * or instantiated.
 */
public class TransformerFactoryConfigurationError extends Error {
    private static final long serialVersionUID = -6527718720676281516L;

    /**
     * <code>Exception</code> for the
     *  <code>TransformerFactoryConfigurationError</code>.
     */
    private Exception exception;

    /**
     * Create a new <code>TransformerFactoryConfigurationError</code> with no
     * detail mesage.
     */
    public TransformerFactoryConfigurationError() {

        super();

        this.exception = null;
    }

    /**
     * Create a new <code>TransformerFactoryConfigurationError</code> with
     * the <code>String</code> specified as an error message.
     *
     * @param msg The error message for the exception.
     */
    public TransformerFactoryConfigurationError(String msg) {

        super(msg);

        this.exception = null;
    }

```

```java
    /**
     * Create a new <code>TransformerFactoryConfigurationError</code> with a
     * given <code>Exception</code> base cause of the error.
     *
     * @param e The exception to be encapsulated in a
     * TransformerFactoryConfigurationError.
     */
    public TransformerFactoryConfigurationError(Exception e) {

        super(e.toString());

        this.exception = e;
    }

    /**
     * Create a new <code>TransformerFactoryConfigurationError</code> with the
     * given <code>Exception</code> base cause and detail message.
     *
     * @param e The exception to be encapsulated in a
     * TransformerFactoryConfigurationError
     * @param msg The detail message.
     */
    public TransformerFactoryConfigurationError(Exception e, String msg) {

        super(msg);

        this.exception = e;
    }

    /**
     * Return the message (if any) for this error . If there is no
     * message for the exception and there is an encapsulated
     * exception then the message of that exception will be returned.
     *
     * @return The error message.
     */
    public String getMessage() {

        String message = super.getMessage();

        if ((message == null) && (exception != null)) {
            return exception.getMessage();
        }

        return message;
    }

    /**
     * Return the actual exception (if any) that caused this exception to
     * be raised.
     *
     * @return The encapsulated exception, or null if there is none.
     */
```

```
    public Exception getException() {
        return exception;
    }
    /**
     * use the exception chaining mechanism of JDK1.4
    */
    @Override
    public Throwable getCause() {
        return exception;
    }
}
```

21.14 URIResolver.java

Secuencia 238: javax/xml/transform/URIResolver.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform;

/**
 * <p>An object that implements this interface that can be called by the processor
 * to turn a URI used in document(), xsl:import, or xsl:include into a Source object.
 */
public interface URIResolver {

    /**
     * Called by the processor when it encounters
     * an xsl:include, xsl:import, or document() function.
     *
     * @param href An href attribute, which may be relative or absolute.
```

```
     * @param base The base URI against which the first argument will be made
     * absolute if the absolute URI is required.
     *
     * @return A Source object, or null if the href cannot be resolved,
     * and the processor should try to resolve the URI itself.
     *
     * @throws TransformerException if an error occurs when trying to
     * resolve the URI.
     */
    public Source resolve(String href, String base)
        throws TransformerException;
}
```

22 Xml Transform Dom (javax.xml.transform.dom package)

22.1 DOMLocator.java

Secuencia 239: javax/xml/transform/dom/DOMLocator.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.dom;

import javax.xml.transform.SourceLocator;

import org.w3c.dom.Node;


/**
 * Indicates the position of a node in a source DOM, intended
 * primarily for error reporting.  To use a DOMLocator, the receiver of an
```

```
 * error must downcast the {@link javax.xml.transform.SourceLocator}
 * object returned by an exception. A {@link javax.xml.transform.Transformer}
 * may use this object for purposes other than error reporting, for instance,
 * to indicate the source node that originated a result node.
 */
public interface DOMLocator extends SourceLocator {

    /**
     * Return the node where the event occurred.
     *
     * @return The node that is the location for the event.
     */
    public Node getOriginatingNode();
}
```

22.2 DOMResult.java

Secuencia 240: javax/xml/transform/dom/DOMResult.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.dom;

import javax.xml.transform.Result;
import org.w3c.dom.Node;

/**
 * <p>Acts as a holder for a transformation result tree in the form of a Document Object Model (DOM
     ) tree.</p>
 *
```

```
 * <p>If no output DOM source is set, the transformation will create a Document node as the holder
     for the result of the transformation,
 * which may be retrieved with {@link #getNode()}.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public class DOMResult implements Result {

    /** <p>If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns <code>true</code> when passed this value as an argument,
     * the <code>Transformer</code> supports <code>Result</code> output of this type.</p>
     */
    public static final String FEATURE = "http://javax.xml.transform.dom.DOMResult/feature";

    /**
     * <p>Zero-argument default constructor.</p>
     *
     * <p><code>node</code>,
     * <code>siblingNode</code> and
     * <code>systemId</code>
     * will be set to <code>null</code>.</p>
     */
    public DOMResult() {
        setNode(null);
        setNextSibling(null);
        setSystemId(null);
    }

    /**
     * <p>Use a DOM node to create a new output target.</p>
     *
     * <p>In practice, the node should be
     * a {@link org.w3c.dom.Document} node,
     * a {@link org.w3c.dom.DocumentFragment} node, or
     * a {@link org.w3c.dom.Element} node.
     * In other words, a node that accepts children.</p>
     *
     * <p><code>siblingNode</code> and
     * <code>systemId</code>
     * will be set to <code>null</code>.</p>
     *
     * @param node The DOM node that will contain the result tree.
     */
    public DOMResult(Node node) {
        setNode(node);
        setNextSibling(null);
        setSystemId(null);
    }

    /**
     * <p>Use a DOM node to create a new output target with the specified System ID.<p>
     *
     * <p>In practice, the node should be
```

```
     * a {@link org.w3c.dom.Document} node,
     * a {@link org.w3c.dom.DocumentFragment} node, or
     * a {@link org.w3c.dom.Element} node.
     * In other words, a node that accepts children.</p>
     *
     * <p><code>siblingNode</code> will be set to <code>null</code>.</p>
     *
     * @param node The DOM node that will contain the result tree.
     * @param systemId The system identifier which may be used in association with this node.
     */
    public DOMResult(Node node, String systemId) {
        setNode(node);
        setNextSibling(null);
        setSystemId(systemId);
    }

    /**
     * <p>Use a DOM node to create a new output target specifying the child node where the result
       nodes should be inserted before.</p>
     *
     * <p>In practice, <code>node</code> and <code>nextSibling</code> should be
     * a {@link org.w3c.dom.Document} node,
     * a {@link org.w3c.dom.DocumentFragment} node, or
     * a {@link org.w3c.dom.Element} node.
     * In other words, a node that accepts children.</p>
     *
     * <p>Use <code>nextSibling</code> to specify the child node
     * where the result nodes should be inserted before.
     * If <code>nextSibling</code> is not a sibling of <code>node</code>,
     * then an <code>IllegalArgumentException</code> is thrown.
     * If <code>node</code> is <code>null</code> and <code>nextSibling</code> is not <code>null</
       code>,
     * then an <code>IllegalArgumentException</code> is thrown.
     * If <code>nextSibling</code> is <code>null</code>,
     * then the behavior is the same as calling {@link #DOMResult(Node node)},
     * i.e. append the result nodes as the last child of the specified <code>node</code>.</p>
     *
     * <p><code>systemId</code> will be set to <code>null</code>.</p>
     *
     * @param node The DOM node that will contain the result tree.
     * @param nextSibling The child node where the result nodes should be inserted before.
     *
     * @throws IllegalArgumentException If <code>nextSibling</code> is not a sibling of <code>node
       </code> or
     *   <code>node</code> is <code>null</code> and <code>nextSibling</code>
     *   is not <code>null</code>.
     *
     * @since 1.5
     */
    public DOMResult(Node node, Node nextSibling) {

        // does the corrent parent/child relationship exist?
        if (nextSibling != null) {
```

```
            // cannot be a sibling of a null node
            if (node == null) {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling is contained by the \"null\" node.");
            }

            // nextSibling contained by node?
            if ((node.compareDocumentPosition(nextSibling)&Node.DOCUMENT_POSITION_CONTAINED_BY)==0) {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling is not contained by the node.");
            }
        }

        setNode(node);
        setNextSibling(nextSibling);
        setSystemId(null);
    }

    /**
     * <p>Use a DOM node to create a new output target specifying the child node where the result nodes should be inserted before and
     * the specified System ID.</p>
     *
     * <p>In practice, <code>node</code> and <code>nextSibling</code> should be
     * a {@link org.w3c.dom.Document} node,
     * a {@link org.w3c.dom.DocumentFragment} node, or a
     * {@link org.w3c.dom.Element} node.
     * In other words, a node that accepts children.</p>
     *
     * <p>Use <code>nextSibling</code> to specify the child node
     * where the result nodes should be inserted before.
     * If <code>nextSibling</code> is not a sibling of <code>node</code>,
     * then an <code>IllegalArgumentException</code> is thrown.
     * If <code>node</code> is <code>null</code> and <code>nextSibling</code> is not <code>null</code>,
     * then an <code>IllegalArgumentException</code> is thrown.
     * If <code>nextSibling</code> is <code>null</code>,
     * then the behavior is the same as calling {@link #DOMResult(Node node, String systemId)},
     * i.e. append the result nodes as the last child of the specified node and use the specified System ID.</p>
     *
     * @param node The DOM node that will contain the result tree.
     * @param nextSibling The child node where the result nodes should be inserted before.
     * @param systemId The system identifier which may be used in association with this node.
     *
     * @throws IllegalArgumentException If <code>nextSibling</code> is not a
     *   sibling of <code>node</code> or
     *   <code>node</code> is <code>null</code> and <code>nextSibling</code>
     *   is not <code>null</code>.
     *
     * @since 1.5
     */
```

```
    public DOMResult(Node node, Node nextSibling, String systemId) {

        // does the corrent parent/child relationship exist?
        if (nextSibling != null) {
            // cannot be a sibling of a null node
            if (node == null) {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling
                    is contained by the \"null\" node.");
            }

            // nextSibling contained by node?
            if ((node.compareDocumentPosition(nextSibling)&Node.DOCUMENT_POSITION_CONTAINED_BY)==0)
                 {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling
                    is not contained by the node.");
            }
        }

        setNode(node);
        setNextSibling(nextSibling);
        setSystemId(systemId);
    }

    /**
     * <p>Set the node that will contain the result DOM tree.<p>
     *
     * <p>In practice, the node should be
     * a {@link org.w3c.dom.Document} node,
     * a {@link org.w3c.dom.DocumentFragment} node, or
     * a {@link org.w3c.dom.Element} node.
     * In other words, a node that accepts children.</p>
     *
     * <p>An <code>IllegalStateException</code> is thrown if
     * <code>nextSibling</code> is not <code>null</code> and
     * <code>node</code> is not a parent of <code>nextSibling</code>.
     * An <code>IllegalStateException</code> is thrown if <code>node</code> is <code>null</code>
       and
     * <code>nextSibling</code> is not <code>null</code>.</p>
     *
     * @param node The node to which the transformation will be appended.
     *
     * @throws IllegalStateException If <code>nextSibling</code> is not
     *   <code>null</code> and
     *   <code>nextSibling</code> is not a child of <code>node</code> or
     *   <code>node</code> is <code>null</code> and
     *   <code>nextSibling</code> is not <code>null</code>.
     */
    public void setNode(Node node) {
        // does the corrent parent/child relationship exist?
        if (nextSibling != null) {
            // cannot be a sibling of a null node
            if (node == null) {
                throw new IllegalStateException("Cannot create a DOMResult when the nextSibling is
```

```
                    contained by the \"null\" node.");
            }

            // nextSibling contained by node?
            if ((node.compareDocumentPosition(nextSibling)&Node.DOCUMENT_POSITION_CONTAINED_BY)==0) {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling is not contained by the node.");
            }
        }

        this.node = node;
    }

    /**
     * <p>Get the node that will contain the result DOM tree.</p>
     *
     * <p>If no node was set via
     * {@link #DOMResult(Node node)},
     * {@link #DOMResult(Node node, String systeId)},
     * {@link #DOMResult(Node node, Node nextSibling)},
     * {@link #DOMResult(Node node, Node nextSibling, String systemId)} or
     * {@link #setNode(Node node)},
     * then the node will be set by the transformation, and may be obtained from this method once the transformation is complete.
     * Calling this method before the transformation will return <code>null</code>.</p>
     *
     * @return The node to which the transformation will be appended.
     */
    public Node getNode() {
        return node;
    }

    /**
     * <p>Set the child node before which the result nodes will be inserted.</p>
     *
     * <p>Use <code>nextSibling</code> to specify the child node
     * before which the result nodes should be inserted.
     * If <code>nextSibling</code> is not a descendant of <code>node</code>,
     * then an <code>IllegalArgumentException</code> is thrown.
     * If <code>node</code> is <code>null</code> and <code>nextSibling</code> is not <code>null</code>,
     * then an <code>IllegalStateException</code> is thrown.
     * If <code>nextSibling</code> is <code>null</code>,
     * then the behavior is the same as calling {@link #DOMResult(Node node)},
     * i.e. append the result nodes as the last child of the specified <code>node</code>.</p>
     *
     * @param nextSibling The child node before which the result nodes will be inserted.
     *
     * @throws IllegalArgumentException If <code>nextSibling</code> is not a
     *   descendant of <code>node</code>.
     * @throws IllegalStateException If <code>node</code> is <code>null</code>
     *   and <code>nextSibling</code> is not <code>null</code>.
```

```
     *
     * @since 1.5
     */
    public void setNextSibling(Node nextSibling) {

        // does the corrent parent/child relationship exist?
        if (nextSibling != null) {
            // cannot be a sibling of a null node
            if (node == null) {
                throw new IllegalStateException("Cannot create a DOMResult when the nextSibling is contained by the \"null\" node.");
            }

            // nextSibling contained by node?
            if ((node.compareDocumentPosition(nextSibling)&Node.DOCUMENT_POSITION_CONTAINED_BY)==0) {
                throw new IllegalArgumentException("Cannot create a DOMResult when the nextSibling is not contained by the node.");
            }
        }

        this.nextSibling = nextSibling;
    }

    /**
     * <p>Get the child node before which the result nodes will be inserted.</p>
     *
     * <p>If no node was set via
     * {@link #DOMResult(Node node, Node nextSibling)},
     * {@link #DOMResult(Node node, Node nextSibling, String systemId)} or
     * {@link #setNextSibling(Node nextSibling)},
     * then <code>null</code> will be returned.</p>
     *
     * @return The child node before which the result nodes will be inserted.
     *
     * @since 1.5
     */
    public Node getNextSibling() {
        return nextSibling;
    }

    /**
     * <p>Set the systemId that may be used in association with the node.</p>
     *
     * @param systemId The system identifier as a URI string.
     */
    public void setSystemId(String systemId) {
        this.systemId = systemId;
    }

    /**
     * <p>Get the System Identifier.</p>
     *
```

```
     * <p>If no System ID was set via
     * {@link #DOMResult(Node node, String systemId)},
     * {@link #DOMResult(Node node, Node nextSibling, String systemId)} or
     * {@link #setSystemId(String systemId)},
     * then <code>null</code> will be returned.</p>
     *
     * @return The system identifier.
     */
    public String getSystemId() {
        return systemId;
    }

    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    /**
     * <p>The node to which the transformation will be appended.</p>
     */
    private Node node = null;

    /**
     * <p>The child node before which the result nodes will be inserted.</p>
     *
     * @since 1.5
     */
    private Node nextSibling = null;

    /**
     * <p>The System ID that may be used in association with the node.</p>
     */
    private String systemId = null;
}
```

22.3 DOMSource.java

Secuencia 241: javax/xml/transform/dom/DOMSource.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.dom;

import javax.xml.transform.Source;

import org.w3c.dom.Node;

/**
 * <p>Acts as a holder for a transformation Source tree in the
 * form of a Document Object Model (DOM) tree.</p>
 *
 * <p>Note that XSLT requires namespace support. Attempting to transform a DOM
 * that was not contructed with a namespace-aware parser may result in errors.
 * Parsers can be made namespace aware by calling
 * {@link javax.xml.parsers.DocumentBuilderFactory#setNamespaceAware(boolean awareness)}.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @see <a href="http://www.w3.org/TR/DOM-Level-2">Document Object Model (DOM) Level 2
     Specification</a>
 */
public class DOMSource implements Source {

    /**
     * <p><code>Node</code> to serve as DOM source.</p>
     */
    private Node node;

    /**
     * <p>The base ID (URL or system ID) from where URLs
     * will be resolved.</p>
     */
    private String systemID;

    /** If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the Transformer supports Source input of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.dom.DOMSource/feature";

    /**
     * <p>Zero-argument default constructor.  If this constructor is used, and
     * no DOM source is set using {@link #setNode(Node node)} , then the
     * <code>Transformer</code> will
     * create an empty source {@link org.w3c.dom.Document} using
```

```
     * {@link javax.xml.parsers.DocumentBuilder#newDocument()}.</p>
     *
     * @see javax.xml.transform.Transformer#transform(Source xmlSource, Result outputTarget)
     */
    public DOMSource() { }

    /**
     * Create a new input source with a DOM node.  The operation
     * will be applied to the subtree rooted at this node.  In XSLT,
     * a "/" pattern still means the root of the tree (not the subtree),
     * and the evaluation of global variables and parameters is done
     * from the root node also.
     *
     * @param n The DOM node that will contain the Source tree.
     */
    public DOMSource(Node n) {
        setNode(n);
    }

    /**
     * Create a new input source with a DOM node, and with the
     * system ID also passed in as the base URI.
     *
     * @param node The DOM node that will contain the Source tree.
     * @param systemID Specifies the base URI associated with node.
     */
    public DOMSource(Node node, String systemID) {
        setNode(node);
        setSystemId(systemID);
    }

    /**
     * Set the node that will represents a Source DOM tree.
     *
     * @param node The node that is to be transformed.
     */
    public void setNode(Node node) {
        this.node = node;
    }

    /**
     * Get the node that represents a Source DOM tree.
     *
     * @return The node that is to be transformed.
     */
    public Node getNode() {
        return node;
    }

    /**
     * Set the base ID (URL or system ID) from where URLs
     * will be resolved.
     *
```

```
     * @param systemID Base URL for this DOM tree.
     */
    public void setSystemId(String systemID) {
        this.systemID = systemID;
    }

    /**
     * Get the base ID (URL or system ID) from where URLs
     * will be resolved.
     *
     * @return Base URL for this DOM tree.
     */
    public String getSystemId() {
        return this.systemID;
    }
}
```

23 Xml Transform Sax (javax.xml.transform.sax package)

23.1 SAXResult.java

Secuencia 242: javax/xml/transform/sax/SAXResult.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.sax;

import javax.xml.transform.Result;

import org.xml.sax.ContentHandler;
import org.xml.sax.ext.LexicalHandler;
```

```java

/**
 * <p>Acts as an holder for a transformation Result.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public class SAXResult implements Result {

    /**
     * If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the Transformer supports Result output of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.sax.SAXResult/feature";

    /**
     * Zero-argument default constructor.
     */
    public SAXResult() {
    }

    /**
     * Create a SAXResult that targets a SAX2 {@link org.xml.sax.ContentHandler}.
     *
     * @param handler Must be a non-null ContentHandler reference.
     */
    public SAXResult(ContentHandler handler) {
        setHandler(handler);
    }

    /**
     * Set the target to be a SAX2 {@link org.xml.sax.ContentHandler}.
     *
     * @param handler Must be a non-null ContentHandler reference.
     */
    public void setHandler(ContentHandler handler) {
        this.handler = handler;
    }

    /**
     * Get the {@link org.xml.sax.ContentHandler} that is the Result.
     *
     * @return The ContentHandler that is to be transformation output.
     */
    public ContentHandler getHandler() {
        return handler;
    }

    /**
     * Set the SAX2 {@link org.xml.sax.ext.LexicalHandler} for the output.
     *
     * <p>This is needed to handle XML comments and the like.  If the
```

```
     * lexical handler is not set, an attempt should be made by the
     * transformer to cast the {@link org.xml.sax.ContentHandler} to a
     * <code>LexicalHandler</code>.</p>
     *
     * @param handler A non-null <code>LexicalHandler</code> for
     * handling lexical parse events.
     */
    public void setLexicalHandler(LexicalHandler handler) {
        this.lexhandler = handler;
    }

    /**
     * Get a SAX2 {@link org.xml.sax.ext.LexicalHandler} for the output.
     *
     * @return A <code>LexicalHandler</code>, or null.
     */
    public LexicalHandler getLexicalHandler() {
        return lexhandler;
    }

    /**
     * Method setSystemId Set the systemID that may be used in association
     * with the {@link org.xml.sax.ContentHandler}.
     *
     * @param systemId The system identifier as a URI string.
     */
    public void setSystemId(String systemId) {
        this.systemId = systemId;
    }

    /**
     * Get the system identifier that was set with setSystemId.
     *
     * @return The system identifier that was set with setSystemId, or null
     * if setSystemId was not called.
     */
    public String getSystemId() {
        return systemId;
    }

    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    /**
     * The handler for parse events.
     */
    private ContentHandler handler;

    /**
     * The handler for lexical events.
     */
    private LexicalHandler lexhandler;
```

```java

    /**
     * The systemID that may be used in association
     * with the node.
     */
    private String systemId;
}
```

23.2 SAXSource.java

Secuencia 243: javax/xml/transform/sax/SAXSource.java

```java
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.sax;

import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamSource;

import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

/**
 * <p>Acts as an holder for SAX-style Source.</p>
 *
 * <p>Note that XSLT requires namespace support. Attempting to transform an
 * input source that is not
 * generated with a namespace-aware parser may result in errors.
 * Parsers can be made namespace aware by calling the
 * {@link javax.xml.parsers.SAXParserFactory#setNamespaceAware(boolean awareness)} method.</p>
 *
```

```
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public class SAXSource implements Source {

    /**
     * If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the Transformer supports Source input of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.sax.SAXSource/feature";

    /**
     * <p>Zero-argument default constructor.  If this constructor is used, and
     * no SAX source is set using
     * {@link #setInputSource(InputSource inputSource)} , then the
     * <code>Transformer</code> will
     * create an empty source {@link org.xml.sax.InputSource} using
     * {@link org.xml.sax.InputSource#InputSource() new InputSource()}.</p>
     *
     * @see javax.xml.transform.Transformer#transform(Source xmlSource, Result outputTarget)
     */
    public SAXSource() { }

    /**
     * Create a <code>SAXSource</code>, using an {@link org.xml.sax.XMLReader}
     * and a SAX InputSource. The {@link javax.xml.transform.Transformer}
     * or {@link javax.xml.transform.sax.SAXTransformerFactory} will set itself
     * to be the reader's {@link org.xml.sax.ContentHandler}, and then will call
     * reader.parse(inputSource).
     *
     * @param reader An XMLReader to be used for the parse.
     * @param inputSource A SAX input source reference that must be non-null
     * and that will be passed to the reader parse method.
     */
    public SAXSource(XMLReader reader, InputSource inputSource) {
        this.reader      = reader;
        this.inputSource = inputSource;
    }

    /**
     * Create a <code>SAXSource</code>, using a SAX <code>InputSource</code>.
     * The {@link javax.xml.transform.Transformer} or
     * {@link javax.xml.transform.sax.SAXTransformerFactory} creates a
     * reader via {@link org.xml.sax.helpers.XMLReaderFactory}
     * (if setXMLReader is not used), sets itself as
     * the reader's {@link org.xml.sax.ContentHandler}, and calls
     * reader.parse(inputSource).
     *
     * @param inputSource An input source reference that must be non-null
     * and that will be passed to the parse method of the reader.
     */
    public SAXSource(InputSource inputSource) {
```

```
        this.inputSource = inputSource;
    }

    /**
     * Set the XMLReader to be used for the Source.
     *
     * @param reader A valid XMLReader or XMLFilter reference.
     */
    public void setXMLReader(XMLReader reader) {
        this.reader = reader;
    }

    /**
     * Get the XMLReader to be used for the Source.
     *
     * @return A valid XMLReader or XMLFilter reference, or null.
     */
    public XMLReader getXMLReader() {
        return reader;
    }

    /**
     * Set the SAX InputSource to be used for the Source.
     *
     * @param inputSource A valid InputSource reference.
     */
    public void setInputSource(InputSource inputSource) {
        this.inputSource = inputSource;
    }

    /**
     * Get the SAX InputSource to be used for the Source.
     *
     * @return A valid InputSource reference, or null.
     */
    public InputSource getInputSource() {
        return inputSource;
    }

    /**
     * Set the system identifier for this Source.  If an input source
     * has already been set, it will set the system ID or that
     * input source, otherwise it will create a new input source.
     *
     * <p>The system identifier is optional if there is a byte stream
     * or a character stream, but it is still useful to provide one,
     * since the application can use it to resolve relative URIs
     * and can include it in error messages and warnings (the parser
     * will attempt to open a connection to the URI only if
     * no byte stream or character stream is specified).</p>
     *
     * @param systemId The system identifier as a URI string.
     */
```

```
    public void setSystemId(String systemId) {

        if (null == inputSource) {
            inputSource = new InputSource(systemId);
        } else {
            inputSource.setSystemId(systemId);
        }
    }

    /**
     * <p>Get the base ID (URI or system ID) from where URIs
     * will be resolved.</p>
     *
     * @return Base URL for the <code>Source</code>, or <code>null</code>.
     */
    public String getSystemId() {

        if (inputSource == null) {
            return null;
        } else {
            return inputSource.getSystemId();
        }
    }

    /**
     * The XMLReader to be used for the source tree input. May be null.
     */
    private XMLReader reader;

    /**
     * <p>The SAX InputSource to be used for the source tree input.
     * Should not be <code>null</code>.</p>
     */
    private InputSource inputSource;

    /**
     * Attempt to obtain a SAX InputSource object from a Source
     * object.
     *
     * @param source Must be a non-null Source reference.
     *
     * @return An InputSource, or null if Source can not be converted.
     */
    public static InputSource sourceToInputSource(Source source) {

        if (source instanceof SAXSource) {
            return ((SAXSource) source).getInputSource();
        } else if (source instanceof StreamSource) {
            StreamSource ss      = (StreamSource) source;
            InputSource  isource = new InputSource(ss.getSystemId());

            isource.setByteStream(ss.getInputStream());
            isource.setCharacterStream(ss.getReader());
```

```
            isource.setPublicId(ss.getPublicId());

            return isource;
        } else {
            return null;
        }
    }
}
```

23.3 SAXTransformerFactory.java

Secuencia 244: javax/xml/transform/sax/SAXTransformerFactory.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.sax;

import javax.xml.transform.*;

import org.xml.sax.XMLFilter;

/**
 * This class extends TransformerFactory to provide SAX-specific
 * factory methods.  It provides two types of ContentHandlers,
 * one for creating Transformers, the other for creating Templates
 * objects.
 *
 * <p>If an application wants to set the ErrorHandler or EntityResolver
 * for an XMLReader used during a transformation, it should use a URIResolver
 * to return the SAXSource which provides (with getXMLReader) a reference to
 * the XMLReader.</p>
```

```
 */
public abstract class SAXTransformerFactory extends TransformerFactory {

    /** If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the TransformerFactory returned from
     * {@link javax.xml.transform.TransformerFactory#newInstance} may
     * be safely cast to a SAXTransformerFactory.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.sax.SAXTransformerFactory/feature";

    /** If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the {@link #newXMLFilter(Source src)}
     * and {@link #newXMLFilter(Templates templates)} methods are supported.
     */
    public static final String FEATURE_XMLFILTER =
        "http://javax.xml.transform.sax.SAXTransformerFactory/feature/xmlfilter";

    /**
     * The default constructor is protected on purpose.
     */
    protected SAXTransformerFactory() {}

    /**
     * Get a TransformerHandler object that can process SAX
     * ContentHandler events into a Result, based on the transformation
     * instructions specified by the argument.
     *
     * @param src The Source of the transformation instructions.
     *
     * @return TransformerHandler ready to transform SAX events.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TransformerHandler can not be created.
     */
    public abstract TransformerHandler newTransformerHandler(Source src)
        throws TransformerConfigurationException;

    /**
     * Get a TransformerHandler object that can process SAX
     * ContentHandler events into a Result, based on the Templates argument.
     *
     * @param templates The compiled transformation instructions.
     *
     * @return TransformerHandler ready to transform SAX events.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TransformerHandler can not be created.
     */
    public abstract TransformerHandler newTransformerHandler(
        Templates templates) throws TransformerConfigurationException;
```

```java

    /**
     * Get a TransformerHandler object that can process SAX
     * ContentHandler events into a Result. The transformation
     * is defined as an identity (or copy) transformation, for example
     * to copy a series of SAX parse events into a DOM tree.
     *
     * @return A non-null reference to a TransformerHandler, that may
     * be used as a ContentHandler for SAX parse events.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TransformerHandler cannot be created.
     */
    public abstract TransformerHandler newTransformerHandler()
        throws TransformerConfigurationException;

    /**
     * Get a TemplatesHandler object that can process SAX
     * ContentHandler events into a Templates object.
     *
     * @return A non-null reference to a TransformerHandler, that may
     * be used as a ContentHandler for SAX parse events.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TemplatesHandler cannot be created.
     */
    public abstract TemplatesHandler newTemplatesHandler()
        throws TransformerConfigurationException;

    /**
     * Create an XMLFilter that uses the given Source as the
     * transformation instructions.
     *
     * @param src The Source of the transformation instructions.
     *
     * @return An XMLFilter object, or null if this feature is not supported.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TemplatesHandler cannot be created.
     */
    public abstract XMLFilter newXMLFilter(Source src)
        throws TransformerConfigurationException;

    /**
     * Create an XMLFilter, based on the Templates argument..
     *
     * @param templates The compiled transformation instructions.
     *
     * @return An XMLFilter object, or null if this feature is not supported.
     *
     * @throws TransformerConfigurationException If for some reason the
     * TemplatesHandler cannot be created.
     */
```

```
    public abstract XMLFilter newXMLFilter(Templates templates)
        throws TransformerConfigurationException;
}
```

23.4 TemplatesHandler.java

Secuencia 245: javax/xml/transform/sax/TemplatesHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.sax;

import javax.xml.transform.*;

import org.xml.sax.ContentHandler;

/**
 * A SAX ContentHandler that may be used to process SAX
 * parse events (parsing transformation instructions) into a Templates object.
 *
 * <p>Note that TemplatesHandler does not need to implement LexicalHandler.</p>
 */
public interface TemplatesHandler extends ContentHandler {

    /**
     * When a TemplatesHandler object is used as a ContentHandler
     * for the parsing of transformation instructions, it creates a Templates object,
     * which the caller can get once the SAX events have been completed.
     *
     * @return The Templates object that was created during
     * the SAX event process, or null if no Templates object has
```

```
     * been created.
     *
     */
    public Templates getTemplates();

    /**
     * Set the base ID (URI or system ID) for the Templates object
     * created by this builder.  This must be set in order to
     * resolve relative URIs in the stylesheet.  This must be
     * called before the startDocument event.
     *
     * @param systemID Base URI for this stylesheet.
     */
    public void setSystemId(String systemID);

    /**
     * Get the base ID (URI or system ID) from where relative
     * URLs will be resolved.
     * @return The systemID that was set with {@link #setSystemId}.
     */
    public String getSystemId();
}
```

23.5 TransformerHandler.java

Secuencia 246: javax/xml/transform/sax/TransformerHandler.java

```
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.sax;

```

```java
import javax.xml.transform.Result;
import javax.xml.transform.Transformer;

import org.xml.sax.ContentHandler;
import org.xml.sax.DTDHandler;
import org.xml.sax.ext.LexicalHandler;

/**
 * A TransformerHandler
 * listens for SAX ContentHandler parse events and transforms
 * them to a Result.
 */
public interface TransformerHandler
    extends ContentHandler, LexicalHandler, DTDHandler {

    /**
     * <p>Set  the <code>Result</code> associated with this
     * <code>TransformerHandler</code> to be used for the transformation.</p>
     *
     * @param result A <code>Result</code> instance, should not be
     *   <code>null</code>.
     *
     * @throws IllegalArgumentException if result is invalid for some reason.
     */
    public void setResult(Result result) throws IllegalArgumentException;

    /**
     * Set the base ID (URI or system ID) from where relative
     * URLs will be resolved.
     * @param systemID Base URI for the source tree.
     */
    public void setSystemId(String systemID);

    /**
     * Get the base ID (URI or system ID) from where relative
     * URLs will be resolved.
     * @return The systemID that was set with {@link #setSystemId}.
     */
    public String getSystemId();

    /**
     * <p>Get the <code>Transformer</code> associated with this handler, which
     * is needed in order to set parameters and output properties.</p>
     *
     * @return <code>Transformer</code> associated with this
     *   <code>TransformerHandler</code>.
     */
    public Transformer getTransformer();
}
```

24 Xml Transform Stax (javax.xml.transform.stax package)

24.1 StAXResult.java

Secuencia 247: javax/xml/transform/stax/StAXResult.java

```
/*
 * Copyright (c) 2005, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.stax;

import javax.xml.stream.XMLEventWriter;
import javax.xml.stream.XMLStreamWriter;
import javax.xml.transform.Result;

/**
 * <p>Acts as a holder for an XML {@link Result} in the
 * form of a StAX writer,i.e.
 * {@link XMLStreamWriter} or {@link XMLEventWriter}.
 * <code>StAXResult</code> can be used in all cases that accept
 * a <code>Result</code>, e.g. {@link javax.xml.transform.Transformer},
 * {@link javax.xml.validation.Validator} which accept
 * <code>Result</code> as input.
 *
 * @author <a href="mailto:Neeraj.Bajaj@Sun.com">Neeraj Bajaj</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 *
 * @see <a href="http://jcp.org/en/jsr/detail?id=173">
 *  JSR 173: Streaming API for XML</a>
 * @see XMLStreamWriter
 * @see XMLEventWriter
```

```
 *
 * @since 1.6
 */
public class StAXResult implements Result {
    /** If {@link javax.xml.transform.TransformerFactory#getFeature(String name)}
     * returns true when passed this value as an argument,
     * the Transformer supports Result output of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.stax.StAXResult/feature";

    /**
     * <p><code>XMLEventWriter</code> to be used for
     * <code>Result</code> output.</p>
     */
    private XMLEventWriter xmlEventWriter = null;

    /**
     * <p><code>XMLStreamWriter</code> to be used for
     * <code>Result</code> output.</p>
     */
    private XMLStreamWriter xmlStreamWriter = null;

    /** <p>System identifier for this <code>StAXResult</code>.<p> */
    private String systemId = null;

    /**
     * <p>Creates a new instance of a <code>StAXResult</code>
     * by supplying an {@link XMLEventWriter}.</p>
     *
     * <p><code>XMLEventWriter</code> must be a
     * non-<code>null</code> reference.</p>
     *
     * @param xmlEventWriter <code>XMLEventWriter</code> used to create
     *   this <code>StAXResult</code>.
     *
     * @throws IllegalArgumentException If <code>xmlEventWriter</code> ==
     *   <code>null</code>.
     */
    public StAXResult(final XMLEventWriter xmlEventWriter) {

        if (xmlEventWriter == null) {
            throw new IllegalArgumentException(
                    "StAXResult(XMLEventWriter) with XMLEventWriter == null");
        }

        this.xmlEventWriter = xmlEventWriter;
    }

    /**
     * <p>Creates a new instance of a <code>StAXResult</code>
     * by supplying an {@link XMLStreamWriter}.</p>
     *
```

```
     * <p><code>XMLStreamWriter</code> must be a
     * non-<code>null</code> reference.</p>
     *
     * @param xmlStreamWriter <code>XMLStreamWriter</code> used to create
     *   this <code>StAXResult</code>.
     *
     * @throws IllegalArgumentException If <code>xmlStreamWriter</code> ==
     *   <code>null</code>.
     */
    public StAXResult(final XMLStreamWriter xmlStreamWriter) {

        if (xmlStreamWriter == null) {
            throw new IllegalArgumentException(
                    "StAXResult(XMLStreamWriter) with XMLStreamWriter == null");
        }

        this.xmlStreamWriter = xmlStreamWriter;
    }

    /**
     * <p>Get the <code>XMLEventWriter</code> used by this
     * <code>StAXResult</code>.</p>
     *
     * <p><code>XMLEventWriter</code> will be <code>null</code>
     * if this <code>StAXResult</code> was created with a
     * <code>XMLStreamWriter</code>.</p>
     *
     * @return <code>XMLEventWriter</code> used by this
     *   <code>StAXResult</code>.
     */
    public XMLEventWriter getXMLEventWriter() {

        return xmlEventWriter;
    }

    /**
     * <p>Get the <code>XMLStreamWriter</code> used by this
     * <code>StAXResult</code>.</p>
     *
     * <p><code>XMLStreamWriter</code> will be <code>null</code>
     * if this <code>StAXResult</code> was created with a
     * <code>XMLEventWriter</code>.</p>
     *
     * @return <code>XMLStreamWriter</code> used by this
     *   <code>StAXResult</code>.
     */
    public XMLStreamWriter getXMLStreamWriter() {

        return xmlStreamWriter;
    }

    /**
     * <p>In the context of a <code>StAXResult</code>, it is not appropriate
```

```
     * to explicitly set the system identifier.
     * The <code>XMLEventWriter</code> or <code>XMLStreamWriter</code>
     * used to construct this <code>StAXResult</code> determines the
     * system identifier of the XML result.</p>
     *
     * <p>An {@link UnsupportedOperationException} is <strong>always</strong>
     * thrown by this method.</p>
     *
     * @param systemId Ignored.
     *
     * @throws UnsupportedOperationException Is <strong>always</strong>
     *   thrown by this method.
     */
    public void setSystemId(final String systemId) {

        throw new UnsupportedOperationException(
                "StAXResult#setSystemId(systemId) cannot set the "
                + "system identifier for a StAXResult");
    }

    /**
     * <p>The returned system identifier is always <code>null</code>.</p>
     *
     * @return The returned system identifier is always <code>null</code>.
     */
    public String getSystemId() {

        return null;
    }
}
```

24.2 StAXSource.java

Secuencia 248: javax/xml/transform/stax/StAXSource.java

```
/*
 * Copyright (c) 2005, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

package javax.xml.transform.stax;

import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.events.XMLEvent;
import javax.xml.transform.Source;

/**
 * <p>Acts as a holder for an XML {@link Source} in the
 * form of a StAX reader,i.e.
 * {@link XMLStreamReader} or {@link XMLEventReader}.
 * <code>StAXSource</code> can be used in all cases that accept
 * a <code>Source</code>, e.g. {@link javax.xml.transform.Transformer},
 * {@link javax.xml.validation.Validator} which accept
 * <code>Source</code> as input.
 *
 * <p><code>StAXSource</code>s are consumed during processing
 * and are not reusable.</p>
 *
 * @author <a href="mailto:Neeraj.Bajaj@Sun.com">Neeraj Bajaj</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 *
 * @see <a href="http://jcp.org/en/jsr/detail?id=173">
 *  JSR 173: Streaming API for XML</a>
 * @see XMLStreamReader
 * @see XMLEventReader
 *
 * @since 1.6
 */
public class StAXSource implements Source {

    /** If {@link javax.xml.transform.TransformerFactory#getFeature(String name)}
     * returns true when passed this value as an argument,
     * the Transformer supports Source input of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.stax.StAXSource/feature";

    /** <p><code>XMLEventReader</code> to be used for source input.</p> */
    private XMLEventReader xmlEventReader = null;

    /** <p><code>XMLStreamReader</code> to be used for source input.</p> */
    private XMLStreamReader xmlStreamReader = null;

    /** <p>System identifier of source input.</p> */
```

```
    private String systemId = null;

    /**
     * <p>Creates a new instance of a <code>StAXSource</code>
     * by supplying an {@link XMLEventReader}.</p>
     *
     * <p><code>XMLEventReader</code> must be a
     * non-<code>null</code> reference.</p>
     *
     * <p><code>XMLEventReader</code> must be in
     * {@link XMLStreamConstants#START_DOCUMENT} or
     * {@link XMLStreamConstants#START_ELEMENT} state.</p>
     *
     * @param xmlEventReader <code>XMLEventReader</code> used to create
     *   this <code>StAXSource</code>.
     *
     * @throws XMLStreamException If <code>xmlEventReader</code> access
     *   throws an <code>Exception</code>.
     * @throws IllegalArgumentException If <code>xmlEventReader</code> ==
     *   <code>null</code>.
     * @throws IllegalStateException If <code>xmlEventReader</code>
     *   is not in <code>XMLStreamConstants.START_DOCUMENT</code> or
     *   <code>XMLStreamConstants.START_ELEMENT</code> state.
     */
    public StAXSource(final XMLEventReader xmlEventReader)
        throws XMLStreamException {

        if (xmlEventReader == null) {
            throw new IllegalArgumentException(
                    "StAXSource(XMLEventReader) with XMLEventReader == null");
        }

        // TODO: This is ugly ...
        // there is no way to know the current position(event) of
        // XMLEventReader.  peek() is the only way to know the next event.
        // The next event on the input stream should be
        // XMLStreamConstants.START_DOCUMENT or
        // XMLStreamConstants.START_ELEMENT.
        XMLEvent event = xmlEventReader.peek();
        int eventType = event.getEventType();
        if (eventType != XMLStreamConstants.START_DOCUMENT
                && eventType != XMLStreamConstants.START_ELEMENT) {
            throw new IllegalStateException(
                "StAXSource(XMLEventReader) with XMLEventReader "
                + "not in XMLStreamConstants.START_DOCUMENT or "
                + "XMLStreamConstants.START_ELEMENT state");
        }

        this.xmlEventReader = xmlEventReader;
        systemId = event.getLocation().getSystemId();
    }

    /**
```

```
     * <p>Creates a new instance of a <code>StAXSource</code>
     * by supplying an {@link XMLStreamReader}.</p>
     *
     * <p><code>XMLStreamReader</code> must be a
     * non-<code>null</code> reference.</p>
     *
     * <p><code>XMLStreamReader</code> must be in
     * {@link XMLStreamConstants#START_DOCUMENT} or
     * {@link XMLStreamConstants#START_ELEMENT} state.</p>
     *
     * @param xmlStreamReader <code>XMLStreamReader</code> used to create
     *   this <code>StAXSource</code>.
     *
     * @throws IllegalArgumentException If <code>xmlStreamReader</code> ==
     *   <code>null</code>.
     * @throws IllegalStateException If <code>xmlStreamReader</code>
     *   is not in <code>XMLStreamConstants.START_DOCUMENT</code> or
     *   <code>XMLStreamConstants.START_ELEMENT</code> state.
     */
    public StAXSource(final XMLStreamReader xmlStreamReader) {

        if (xmlStreamReader == null) {
            throw new IllegalArgumentException(
                    "StAXSource(XMLStreamReader) with XMLStreamReader == null");
        }

        int eventType = xmlStreamReader.getEventType();
        if (eventType != XMLStreamConstants.START_DOCUMENT
                && eventType != XMLStreamConstants.START_ELEMENT) {
            throw new IllegalStateException(
                    "StAXSource(XMLStreamReader) with XMLStreamReader"
                    + "not in XMLStreamConstants.START_DOCUMENT or "
                    + "XMLStreamConstants.START_ELEMENT state");
        }

        this.xmlStreamReader = xmlStreamReader;
        systemId = xmlStreamReader.getLocation().getSystemId();
    }

    /**
     * <p>Get the <code>XMLEventReader</code> used by this
     * <code>StAXSource</code>.</p>
     *
     * <p><code>XMLEventReader</code> will be <code>null</code>.
     * if this <code>StAXSource</code> was created with a
     * <code>XMLStreamReader</code>.</p>
     *
     * @return <code>XMLEventReader</code> used by this
     *   <code>StAXSource</code>.
     */
    public XMLEventReader getXMLEventReader() {

        return xmlEventReader;
```

```
    }

    /**
     * <p>Get the <code>XMLStreamReader</code> used by this
     * <code>StAXSource</code>.</p>
     *
     * <p><code>XMLStreamReader</code> will be <code>null</code>
     * if this <code>StAXSource</code> was created with a
     * <code>XMLEventReader</code>.</p>
     *
     * @return <code>XMLStreamReader</code> used by this
     *   <code>StAXSource</code>.
     */
    public XMLStreamReader getXMLStreamReader() {

        return xmlStreamReader;
    }

    /**
     * <p>In the context of a <code>StAXSource</code>, it is not appropriate
     * to explicitly set the system identifier.
     * The <code>XMLStreamReader</code> or <code>XMLEventReader</code>
     * used to construct this <code>StAXSource</code> determines the
     * system identifier of the XML source.</p>
     *
     * <p>An {@link UnsupportedOperationException} is <strong>always</strong>
     * thrown by this method.</p>
     *
     * @param systemId Ignored.
     *
     * @throws UnsupportedOperationException Is <strong>always</strong>
     *   thrown by this method.
     */
    public void setSystemId(final String systemId) {

        throw new UnsupportedOperationException(
                "StAXSource#setSystemId(systemId) cannot set the "
                + "system identifier for a StAXSource");
    }

    /**
     * <p>Get the system identifier used by this
     * <code>StAXSource</code>.</p>
     *
     * <p>The <code>XMLStreamReader</code> or <code>XMLEventReader</code>
     * used to construct this <code>StAXSource</code> is queried to determine
     * the system identifier of the XML source.</p>
     *
     * <p>The system identifier may be <code>null</code> or
     * an empty <code>""</code> <code>String</code>.</p>
     *
     * @return System identifier used by this <code>StAXSource</code>.
     */
```

```
    public String getSystemId() {

        return systemId;
    }
}
```

25 Xml Transform Stream (javax.xml.transform.stream package)

25.1 StreamResult.java

Secuencia 249: javax/xml/transform/stream/StreamResult.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.stream;

import javax.xml.transform.Result;

import java.io.File;
import java.io.OutputStream;
import java.io.Writer;
import java.net.MalformedURLException;

/**
 * <p>Acts as an holder for a transformation result,
 * which may be XML, plain Text, HTML, or some other form of markup.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public class StreamResult implements Result {

```

```
    /** If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
     * the Transformer supports Result output of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.stream.StreamResult/feature";

    /**
     * Zero-argument default constructor.
     */
    public StreamResult() {
    }

    /**
     * Construct a StreamResult from a byte stream.  Normally,
     * a stream should be used rather than a reader, so that
     * the transformer may use instructions contained in the
     * transformation instructions to control the encoding.
     *
     * @param outputStream A valid OutputStream reference.
     */
    public StreamResult(OutputStream outputStream) {
        setOutputStream(outputStream);
    }

    /**
     * Construct a StreamResult from a character stream.  Normally,
     * a stream should be used rather than a reader, so that
     * the transformer may use instructions contained in the
     * transformation instructions to control the encoding.  However,
     * there are times when it is useful to write to a character
     * stream, such as when using a StringWriter.
     *
     * @param writer  A valid Writer reference.
     */
    public StreamResult(Writer writer) {
        setWriter(writer);
    }

    /**
     * Construct a StreamResult from a URL.
     *
     * @param systemId Must be a String that conforms to the URI syntax.
     */
    public StreamResult(String systemId) {
        this.systemId = systemId;
    }

    /**
     * Construct a StreamResult from a File.
     *
     * @param f Must a non-null File reference.
     */
```

```java
    public StreamResult(File f) {
        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        setSystemId(f.toURI().toASCIIString());
    }

    /**
     * Set the ByteStream that is to be written to.  Normally,
     * a stream should be used rather than a reader, so that
     * the transformer may use instructions contained in the
     * transformation instructions to control the encoding.
     *
     * @param outputStream A valid OutputStream reference.
     */
    public void setOutputStream(OutputStream outputStream) {
        this.outputStream = outputStream;
    }

    /**
     * Get the byte stream that was set with setOutputStream.
     *
     * @return The byte stream that was set with setOutputStream, or null
     * if setOutputStream or the ByteStream constructor was not called.
     */
    public OutputStream getOutputStream() {
        return outputStream;
    }

    /**
     * Set the writer that is to receive the result.  Normally,
     * a stream should be used rather than a writer, so that
     * the transformer may use instructions contained in the
     * transformation instructions to control the encoding.  However,
     * there are times when it is useful to write to a writer,
     * such as when using a StringWriter.
     *
     * @param writer  A valid Writer reference.
     */
    public void setWriter(Writer writer) {
        this.writer = writer;
    }

    /**
     * Get the character stream that was set with setWriter.
     *
     * @return The character stream that was set with setWriter, or null
     * if setWriter or the Writer constructor was not called.
     */
    public Writer getWriter() {
        return writer;
    }

```

```
    /**
     * Set the systemID that may be used in association
     * with the byte or character stream, or, if neither is set, use
     * this value as a writeable URI (probably a file name).
     *
     * @param systemId The system identifier as a URI string.
     */
    public void setSystemId(String systemId) {
        this.systemId = systemId;
    }

    /**
     * <p>Set the system ID from a <code>File</code> reference.</p>
     *
     *
     * @param f Must a non-null File reference.
     */
    public void setSystemId(File f) {
        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        this.systemId = f.toURI().toASCIIString();
    }

    /**
     * Get the system identifier that was set with setSystemId.
     *
     * @return The system identifier that was set with setSystemId, or null
     * if setSystemId was not called.
     */
    public String getSystemId() {
        return systemId;
    }

    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    /**
     * The systemID that may be used in association
     * with the byte or character stream, or, if neither is set, use
     * this value as a writeable URI (probably a file name).
     */
    private String systemId;

    /**
     * The byte stream that is to be written to.
     */
    private OutputStream outputStream;

    /**
     * The character stream that is to be written to.
     */
```

```
    private Writer writer;
}
```

25.2 StreamSource.java

Secuencia 250: javax/xml/transform/stream/StreamSource.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.transform.stream;

import java.io.File;
import java.io.InputStream;
import java.io.Reader;

import javax.xml.transform.Source;

/**
 * <p>Acts as an holder for a transformation Source in the form
 * of a stream of XML markup.</p>
 *
 * <p><em>Note:</em> Due to their internal use of either a {@link Reader} or {@link InputStream}
     instance,
 * <code>StreamSource</code> instances may only be used once.</p>
 *
 * @author <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 */
public class StreamSource implements Source {

    /** If {@link javax.xml.transform.TransformerFactory#getFeature}
     * returns true when passed this value as an argument,
```

```
     * the Transformer supports Source input of this type.
     */
    public static final String FEATURE =
        "http://javax.xml.transform.stream.StreamSource/feature";

    /**
     * <p>Zero-argument default constructor.  If this constructor is used, and
     * no Stream source is set using
     * {@link #setInputStream(java.io.InputStream inputStream)} or
     * {@link #setReader(java.io.Reader reader)}, then the
     * <code>Transformer</code> will
     * create an empty source {@link java.io.InputStream} using
     * {@link java.io.InputStream#InputStream() new InputStream()}.</p>
     *
     * @see javax.xml.transform.Transformer#transform(Source xmlSource, Result outputTarget)
     */
    public StreamSource() { }

    /**
     * Construct a StreamSource from a byte stream.  Normally,
     * a stream should be used rather than a reader, so
     * the XML parser can resolve character encoding specified
     * by the XML declaration.
     *
     * <p>If this constructor is used to process a stylesheet, normally
     * setSystemId should also be called, so that relative URI references
     * can be resolved.</p>
     *
     * @param inputStream A valid InputStream reference to an XML stream.
     */
    public StreamSource(InputStream inputStream) {
        setInputStream(inputStream);
    }

    /**
     * Construct a StreamSource from a byte stream.  Normally,
     * a stream should be used rather than a reader, so that
     * the XML parser can resolve character encoding specified
     * by the XML declaration.
     *
     * <p>This constructor allows the systemID to be set in addition
     * to the input stream, which allows relative URIs
     * to be processed.</p>
     *
     * @param inputStream A valid InputStream reference to an XML stream.
     * @param systemId Must be a String that conforms to the URI syntax.
     */
    public StreamSource(InputStream inputStream, String systemId) {
        setInputStream(inputStream);
        setSystemId(systemId);
    }

    /**
```

```
     * Construct a StreamSource from a character reader.  Normally,
     * a stream should be used rather than a reader, so that
     * the XML parser can resolve character encoding specified
     * by the XML declaration.  However, in many cases the encoding
     * of the input stream is already resolved, as in the case of
     * reading XML from a StringReader.
     *
     * @param reader A valid Reader reference to an XML character stream.
     */
    public StreamSource(Reader reader) {
        setReader(reader);
    }

    /**
     * Construct a StreamSource from a character reader.  Normally,
     * a stream should be used rather than a reader, so that
     * the XML parser may resolve character encoding specified
     * by the XML declaration.  However, in many cases the encoding
     * of the input stream is already resolved, as in the case of
     * reading XML from a StringReader.
     *
     * @param reader A valid Reader reference to an XML character stream.
     * @param systemId Must be a String that conforms to the URI syntax.
     */
    public StreamSource(Reader reader, String systemId) {
        setReader(reader);
        setSystemId(systemId);
    }

    /**
     * Construct a StreamSource from a URL.
     *
     * @param systemId Must be a String that conforms to the URI syntax.
     */
    public StreamSource(String systemId) {
        this.systemId = systemId;
    }

    /**
     * Construct a StreamSource from a File.
     *
     * @param f Must a non-null File reference.
     */
    public StreamSource(File f) {
        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        setSystemId(f.toURI().toASCIIString());
    }

    /**
     * Set the byte stream to be used as input.  Normally,
     * a stream should be used rather than a reader, so that
```

```
     * the XML parser can resolve character encoding specified
     * by the XML declaration.
     *
     * <p>If this Source object is used to process a stylesheet, normally
     * setSystemId should also be called, so that relative URL references
     * can be resolved.</p>
     *
     * @param inputStream A valid InputStream reference to an XML stream.
     */
    public void setInputStream(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    /**
     * Get the byte stream that was set with setByteStream.
     *
     * @return The byte stream that was set with setByteStream, or null
     * if setByteStream or the ByteStream constructor was not called.
     */
    public InputStream getInputStream() {
        return inputStream;
    }

    /**
     * Set the input to be a character reader.  Normally,
     * a stream should be used rather than a reader, so that
     * the XML parser can resolve character encoding specified
     * by the XML declaration.  However, in many cases the encoding
     * of the input stream is already resolved, as in the case of
     * reading XML from a StringReader.
     *
     * @param reader A valid Reader reference to an XML CharacterStream.
     */
    public void setReader(Reader reader) {
        this.reader = reader;
    }

    /**
     * Get the character stream that was set with setReader.
     *
     * @return The character stream that was set with setReader, or null
     * if setReader or the Reader constructor was not called.
     */
    public Reader getReader() {
        return reader;
    }

    /**
     * Set the public identifier for this Source.
     *
     * <p>The public identifier is always optional: if the application
     * writer includes one, it will be provided as part of the
     * location information.</p>
```

```
     *
     * @param publicId The public identifier as a string.
     */
    public void setPublicId(String publicId) {
        this.publicId = publicId;
    }

    /**
     * Get the public identifier that was set with setPublicId.
     *
     * @return The public identifier that was set with setPublicId, or null
     * if setPublicId was not called.
     */
    public String getPublicId() {
        return publicId;
    }

    /**
     * Set the system identifier for this Source.
     *
     * <p>The system identifier is optional if there is a byte stream
     * or a character stream, but it is still useful to provide one,
     * since the application can use it to resolve relative URIs
     * and can include it in error messages and warnings (the parser
     * will attempt to open a connection to the URI only if
     * there is no byte stream or character stream specified).</p>
     *
     * @param systemId The system identifier as a URL string.
     */
    public void setSystemId(String systemId) {
        this.systemId = systemId;
    }

    /**
     * Get the system identifier that was set with setSystemId.
     *
     * @return The system identifier that was set with setSystemId, or null
     * if setSystemId was not called.
     */
    public String getSystemId() {
        return systemId;
    }

    /**
     * Set the system ID from a File reference.
     *
     * @param f Must a non-null File reference.
     */
    public void setSystemId(File f) {
        //convert file to appropriate URI, f.toURI().toASCIIString()
        //converts the URI to string as per rule specified in
        //RFC 2396,
        this.systemId = f.toURI().toASCIIString();
```

```
    }

    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    /**
     * The public identifier for this input source, or null.
     */
    private String publicId;

    /**
     * The system identifier as a URL string, or null.
     */
    private String systemId;

    /**
     * The byte stream for this Source, or null.
     */
    private InputStream inputStream;

    /**
     * The character stream for this Source, or null.
     */
    private Reader reader;
}
```

26 Xml Validation (javax.xml.validation package)

26.1 Schema.java

Secuencia 251: javax/xml/validation/Schema.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

package javax.xml.validation;

/**
 * Immutable in-memory representation of grammar.
 *
 * <p>
 * This object represents a set of constraints that can be checked/
 * enforced against an XML document.
 *
 * <p>
 * A {@link Schema} object is thread safe and applications are
 * encouraged to share it across many parsers in many threads.
 *
 * <p>
 * A {@link Schema} object is immutable in the sense that it shouldn't
 * change the set of constraints once it is created. In other words,
 * if an application validates the same document twice against the same
 * {@link Schema}, it must always produce the same result.
 *
 * <p>
 * A {@link Schema} object is usually created from {@link SchemaFactory}.
 *
 * <p>
 * Two kinds of validators can be created from a {@link Schema} object.
 * One is {@link Validator}, which provides highly-level validation
 * operations that cover typical use cases. The other is
 * {@link ValidatorHandler}, which works on top of SAX for better
 * modularity.
 *
 * <p>
 * This specification does not refine
 * the {@link java.lang.Object#equals(java.lang.Object)} method.
 * In other words, if you parse the same schema twice, you may
 * still get <code>!schemaA.equals(schemaB)</code>.
 *
 * @author <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @see <a href="http://www.w3.org/TR/xmlschema-1/">XML Schema Part 1: Structures</a>
 * @see <a href="http://www.w3.org/TR/xml11/">Extensible Markup Language (XML) 1.1</a>
 * @see <a href="http://www.w3.org/TR/REC-xml">Extensible Markup Language (XML) 1.0 (Second Edition
     )</a>
 * @since 1.5
 */
public abstract class Schema {

    /**
     * Constructor for the derived class.
     *
     * <p>
     * The constructor does nothing.
```

```
     */
    protected Schema() {
    }

    /**
     * Creates a new {@link Validator} for this {@link Schema}.
     *
     * <p>A validator enforces/checks the set of constraints this object
     * represents.</p>
     *
     * <p>Implementors should assure that the properties set on the
     * {@link SchemaFactory} that created this {@link Schema} are also
     * set on the {@link Validator} constructed.</p>
     *
     * @return
     *      Always return a non-null valid object.
     */
    public abstract Validator newValidator();

    /**
     * Creates a new {@link ValidatorHandler} for this {@link Schema}.
     *
     * <p>Implementors should assure that the properties set on the
     * {@link SchemaFactory} that created this {@link Schema} are also
     * set on the {@link ValidatorHandler} constructed.</p>
     *
     * @return
     *      Always return a non-null valid object.
     */
    public abstract ValidatorHandler newValidatorHandler();
}
```

26.2 SchemaFactory.java

Secuencia 252: javax/xml/validation/SchemaFactory.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import java.io.File;
import java.net.URL;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamSource;
import org.w3c.dom.ls.LSResourceResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;
import org.xml.sax.SAXParseException;

/**
 * Factory that creates {@link Schema} objects&#x2E; Entry-point to
 * the validation API.
 *
 * <p>
 * {@link SchemaFactory} is a schema compiler. It reads external
 * representations of schemas and prepares them for validation.
 *
 * <p>
 * The {@link SchemaFactory} class is not thread-safe. In other words,
 * it is the application's responsibility to ensure that at most
 * one thread is using a {@link SchemaFactory} object at any
 * given moment. Implementations are encouraged to mark methods
 * as <code>synchronized</code> to protect themselves from broken clients.
 *
 * <p>
 * {@link SchemaFactory} is not re-entrant. While one of the
 * <code>newSchema</code> methods is being invoked, applications
 * may not attempt to recursively invoke the <code>newSchema</code> method,
 * even from the same thread.
 *
 * <h2><a name="schemaLanguage"></a>Schema Language</h2>
 * <p>
 * This spec uses a namespace URI to designate a schema language.
 * The following table shows the values defined by this specification.
 * <p>
 * To be compliant with the spec, the implementation
 * is only required to support W3C XML Schema 1.0. However,
 * if it chooses to support other schema languages listed here,
 * it must conform to the relevant behaviors described in this spec.
 *
 * <p>
 * Schema languages not listed here are expected to
```

```
 * introduce their own URIs to represent themselves.
 * The {@link SchemaFactory} class is capable of locating other
 * implementations for other schema languages at run-time.
 *
 * <p>
 * Note that because the XML DTD is strongly tied to the parsing process
 * and has a significant effect on the parsing process, it is impossible
 * to define the DTD validation as a process independent from parsing.
 * For this reason, this specification does not define the semantics for
 * the XML DTD. This doesn't prohibit implementors from implementing it
 * in a way they see fit, but <em>users are warned that any DTD
 * validation implemented on this interface necessarily deviate from
 * the XML DTD semantics as defined in the XML 1.0</em>.
 *
 * <table border="1" cellpadding="2">
 *   <thead>
 *     <tr>
 *       <th>value</th>
 *       <th>language</th>
 *     </tr>
 *   </thead>
 *   <tbody>
 *     <tr>
 *       <td>{@link javax.xml.XMLConstants#W3C_XML_SCHEMA_NS_URI} ("<code>http://www.w3.org/2001/XMLSchema</code>")</td>
 *       <td><a href="http://www.w3.org/TR/xmlschema-1">W3C XML Schema 1.0</a></td>
 *     </tr>
 *     <tr>
 *       <td>{@link javax.xml.XMLConstants#RELAXNG_NS_URI} ("<code>http://relaxng.org/ns/structure/1.0</code>")</td>
 *       <td><a href="http://www.relaxng.org/">RELAX NG 1.0</a></td>
 *     </tr>
 *   </tbody>
 * </table>
 *
 * @author  <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @author  <a href="mailto:Neeraj.Bajaj@sun.com">Neeraj Bajaj</a>
 *
 * @since 1.5
 */
public abstract class SchemaFactory {

    private static SecuritySupport ss = new SecuritySupport();

    /**
     * <p>Constructor for derived classes.</p>
     *
     * <p>The constructor does nothing.</p>
     *
     * <p>Derived classes must create {@link SchemaFactory} objects that have
     * <code>null</code> {@link ErrorHandler} and
     * <code>null</code> {@link LSResourceResolver}.</p>
     */
```

```
    protected SchemaFactory() {
    }

    /**
     * <p>Lookup an implementation of the <code>SchemaFactory</code> that supports the specified
     * schema language and return it.</p>
     *
     * <p>To find a <code>SchemaFactory</code> object for a given schema language,
     * this method looks the following places in the following order
     * where "the class loader" refers to the context class loader:</p>
     * <ol>
     *  <li>
     *     If the system property
     *     <code>"javax.xml.validation.SchemaFactory:<i>schemaLanguage</i>"</code>
     *     is present (where <i>schemaLanguage</i> is the parameter
     *     to this method), then its value is read
     *     as a class name. The method will try to
     *     create a new instance of this class by using the class loader,
     *     and returns it if it is successfully created.
     *   </li>
     *   <li>
     *     <code>$java.home/lib/jaxp.properties</code> is read and
     *     the value associated with the key being the system property above
     *     is looked for. If present, the value is processed just like above.
     *   </li>
     *   <li>
     *   Use the service-provider loading facilities, defined by the
     *   {@link java.util.ServiceLoader} class, to attempt to locate and load an
     *   implementation of the service using the {@linkplain
     *   java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
     *   the service-provider loading facility will use the {@linkplain
     *   java.lang.Thread#getContextClassLoader() current thread's context class loader}
     *   to attempt to load the service. If the context class
     *   loader is null, the {@linkplain
     *   ClassLoader#getSystemClassLoader() system class loader} will be used.
     *   <br>
     *   Each potential service provider is required to implement the method
     *        {@link #isSchemaLanguageSupported(String schemaLanguage)}.
     *   <br>
     *   The first service provider found that supports the specified schema
     *   language is returned.
     *   <br>
     *   In case of {@link java.util.ServiceConfigurationError} a
     *   {@link SchemaFactoryConfigurationError} will be thrown.
     *   </li>
     *   <li>
     *     Platform default <code>SchemaFactory</code> is located
     *     in a implementation specific way. There must be a platform default
     *     <code>SchemaFactory</code> for W3C XML Schema.
     *   </li>
     * </ol>
     *
     * <p>If everything fails, {@link IllegalArgumentException} will be thrown.</p>
```

```
     *
     * <p><strong>Tip for Trouble-shooting:</strong></p>
     * <p>See {@link java.util.Properties#load(java.io.InputStream)} for
     * exactly how a property file is parsed. In particular, colons ':'
     * need to be escaped in a property file, so make sure schema language
     * URIs are properly escaped in it. For example:</p>
     * <pre>
     * http\://www.w3.org/2001/XMLSchema=org.acme.foo.XSSchemaFactory
     * </pre>
     *
     * @param schemaLanguage
     *      Specifies the schema language which the returned
     *      SchemaFactory will understand. See
     *      <a href="#schemaLanguage">the list of available
     *      schema languages</a> for the possible values.
     *
     * @return New instance of a <code>SchemaFactory</code>
     *
     * @throws IllegalArgumentException
     *      If no implementation of the schema language is available.
     * @throws NullPointerException
     *      If the <code>schemaLanguage</code> parameter is null.
     * @throws SchemaFactoryConfigurationError
     *      If a configuration error is encountered.
     *
     * @see #newInstance(String schemaLanguage, String factoryClassName, ClassLoader classLoader)
     */
    public static SchemaFactory newInstance(String schemaLanguage) {
        ClassLoader cl;
        cl = ss.getContextClassLoader();

        if (cl == null) {
            //cl = ClassLoader.getSystemClassLoader();
            //use the current class loader
            cl = SchemaFactory.class.getClassLoader();
        }

        SchemaFactory f = new SchemaFactoryFinder(cl).newFactory(schemaLanguage);
        if (f == null) {
            throw new IllegalArgumentException(
                    "No SchemaFactory"
                    + " that implements the schema language specified by: " + schemaLanguage
                    + " could be loaded");
        }
        return f;
    }

    /**
     * <p>Obtain a new instance of a <code>SchemaFactory</code> from class name. <code>
       SchemaFactory</code>
     * is returned if specified factory class name supports the specified schema language.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
```

```
     * should be loaded.</p>
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @param schemaLanguage Specifies the schema language which the returned
     *                          <code>SchemaFactory</code> will understand. See
     *                          <a href="#schemaLanguage">the list of available
     *                          schema languages</a> for the possible values.
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <
         code>javax.xml.validation.SchemaFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</
         code>
     *                    current <code>Thread</code>'s context classLoader is used to load the
         factory class.
     *
     * @return New instance of a <code>SchemaFactory</code>
     *
     * @throws IllegalArgumentException
     *                  if <code>factoryClassName</code> is <code>null</code>, or
     *                  the factory class cannot be loaded, instantiated or doesn't
     *                  support the schema language specified in <code>schemLanguage</code>
     *                  parameter.
     *
     * @throws NullPointerException
     *      If the <code>schemaLanguage</code> parameter is null.
     *
     * @see #newInstance(String schemaLanguage)
     *
     * @since 1.6
     */
    public static SchemaFactory newInstance(String schemaLanguage, String factoryClassName,
        ClassLoader classLoader){
        ClassLoader cl = classLoader;

        if (cl == null) {
            cl = ss.getContextClassLoader();
        }

        SchemaFactory f = new SchemaFactoryFinder(cl).createInstance(factoryClassName);
        if (f == null) {
            throw new IllegalArgumentException(
                    "Factory " + factoryClassName
                    + " could not be loaded to implement the schema language specified by: " +
```

```
                        schemaLanguage);
        }
        //if this factory supports the given schemalanguage return this factory else thrown exception
        if(f.isSchemaLanguageSupported(schemaLanguage)){
            return f;
        }else{
            throw new IllegalArgumentException(
                    "Factory " + f.getClass().getName()
                    + " does not implement the schema language specified by: " + schemaLanguage);
        }

    }

    /**
     * <p>Is specified schema supported by this <code>SchemaFactory</code>?</p>
     *
     * @param schemaLanguage Specifies the schema language which the returned <code>SchemaFactory</code> will understand.
     *    <code>schemaLanguage</code> must specify a <a href="#schemaLanguage">valid</a> schema language.
     *
     * @return <code>true</code> if <code>SchemaFactory</code> supports <code>schemaLanguage</code>, else <code>false</code>.
     *
     * @throws NullPointerException If <code>schemaLanguage</code> is <code>null</code>.
     * @throws IllegalArgumentException If <code>schemaLanguage.length() == 0</code>
     *   or <code>schemaLanguage</code> does not specify a <a href="#schemaLanguage">valid</a> schema language.
     */
    public abstract boolean isSchemaLanguageSupported(String schemaLanguage);

    /**
     * Look up the value of a feature flag.
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for a {@link SchemaFactory} to recognize a feature name but
     * temporarily be unable to return its value.
     *
     * <p>Implementors are free (and encouraged) to invent their own features,
     * using names built on their own URIs.</p>
     *
     * @param name The feature name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link SchemaFactory} recognizes the feature name but
     *   cannot determine its value at this time.
     * @throws NullPointerException If <code>name</code> is <code>null</code>.
     *
```

```
     * @see #setFeature(String, boolean)
     */
    public boolean getFeature(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
                throw new NullPointerException("the name parameter is null");
        }
        throw new SAXNotRecognizedException(name);
    }

    /**
     * <p>Set a feature for this <code>SchemaFactory</code>,
     * {@link Schema}s created by this factory, and by extension,
     * {@link Validator}s and {@link ValidatorHandler}s created by
     * those {@link Schema}s.
     * </p>
     *
     * <p>Implementors and developers should pay particular attention
     * to how the special {@link Schema} object returned by {@link
     * #newSchema()} is processed. In some cases, for example, when the
     * <code>SchemaFactory</code> and the class actually loading the
     * schema come from different implementations, it may not be possible
     * for <code>SchemaFactory</code> features to be inherited automatically.
     * Developers should
     * make sure that features, such as secure processing, are explicitly
     * set in both places.</p>
     *
     * <p>The feature name is any fully-qualified URI. It is
     * possible for a {@link SchemaFactory} to expose a feature value but
     * to be unable to change the current value.</p>
     *
     * <p>All implementations are required to support the {@link javax.xml.XMLConstants#
         FEATURE_SECURE_PROCESSING} feature.
     * When the feature is:</p>
     * <ul>
     *   <li>
     *     <code>true</code>: the implementation will limit XML processing to conform to
         implementation limits.
     *     Examples include enity expansion limits and XML Schema constructs that would consume
         large amounts of resources.
     *     If XML processing is limited for security reasons, it will be reported via a call to the
          registered
     *    {@link ErrorHandler#fatalError(SAXParseException exception)}.
     *     See {@link #setErrorHandler(ErrorHandler errorHandler)}.
     *   </li>
     *   <li>
     *     <code>false</code>: the implementation will processing XML according to the XML
         specifications without
     *     regard to possible implementation limits.
     *   </li>
     * </ul>
     *
```

```
     * @param name The feature name, which is a non-null fully-qualified URI.
     * @param value The requested value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link SchemaFactory} recognizes the feature name but
     *   cannot set the requested value.
     * @throws NullPointerException If <code>name</code> is <code>null</code>.
     *
     * @see #getFeature(String)
     */
    public void setFeature(String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
                throw new NullPointerException("the name parameter is null");
        }
        throw new SAXNotRecognizedException(name);
    }

    /**
     * Set the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link SchemaFactory} to recognize a property name but
     * to be unable to change the current value.</p>
     *
     * <p>
     * All implementations that implement JAXP 1.5 or newer are required to
     * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} and
     * {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} properties.
     * </p>
     * <ul>
     *   <li>
     *      <p>Access to external DTDs in Schema files is restricted to the protocols
     *      specified by the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property.
     *      If access is denied during the creation of new Schema due to the restriction
     *      of this property, {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link #newSchema(Source)} or {@link #newSchema(File)}
     *      or {@link #newSchema(URL)} or  or {@link #newSchema(Source[])} method.</p>
     *
     *      <p>Access to external DTDs in xml source files is restricted to the protocols
     *      specified by the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} property.
     *      If access is denied during validation due to the restriction
     *      of this property, {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link javax.xml.validation.Validator#validate(Source)} or
     *      {@link javax.xml.validation.Validator#validate(Source, Result)} method.</p>
     *
     *      <p>Access to external reference set by the schemaLocation attribute is
     *      restricted to the protocols specified by the
     *      {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} property.
     *      If access is denied during validation due to the restriction of this property,
```

```
     *      {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link javax.xml.validation.Validator#validate(Source)} or
     *      {@link javax.xml.validation.Validator#validate(Source, Result)} method.</p>
     *
     *      <p>Access to external reference set by the Import
     *      and Include element is restricted to the protocols specified by the
     *      {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} property.
     *      If access is denied during the creation of new Schema due to the restriction
     *      of this property, {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link #newSchema(Source)} or {@link #newSchema(File)}
     *      or {@link #newSchema(URL)} or {@link #newSchema(Source[])} method.</p>
     *   </li>
     * </ul>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
     * @param object The requested value for the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link SchemaFactory} recognizes the property name but
     *   cannot set the requested value.
     * @throws NullPointerException If <code>name</code> is <code>null</code>.
     */
    public void setProperty(String name, Object object)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
                throw new NullPointerException("the name parameter is null");
        }
        throw new SAXNotRecognizedException(name);
    }

    /**
     * Look up the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link SchemaFactory} to recognize a property name but
     * temporarily be unable to return its value.</p>
     *
     * <p>{@link SchemaFactory}s are not required to recognize any specific
     * property names.</p>
     *
     * <p>Implementors are free (and encouraged) to invent their own properties,
     * using names built on their own URIs.</p>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
```

```
     *   XMLReader recognizes the property name but
     *   cannot determine its value at this time.
     * @throws NullPointerException If <code>name</code> is <code>null</code>.
     *
     * @see #setProperty(String, Object)
     */
    public Object getProperty(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
                throw new NullPointerException("the name parameter is null");
        }
        throw new SAXNotRecognizedException(name);
    }

    /**
     * Sets the {@link ErrorHandler} to receive errors encountered
     * during the <code>newSchema</code> method invocation.
     *
     * <p>
     * Error handler can be used to customize the error handling process
     * during schema parsing. When an {@link ErrorHandler} is set,
     * errors found during the parsing of schemas will be first sent
     * to the {@link ErrorHandler}.
     *
     * <p>
     * The error handler can abort the parsing of a schema immediately
     * by throwing {@link SAXException} from the handler. Or for example
     * it can print an error to the screen and try to continue the
     * processing by returning normally from the {@link ErrorHandler}
     *
     * <p>
     * If any {@link Throwable} (or instances of its derived classes)
     * is thrown from an {@link ErrorHandler},
     * the caller of the <code>newSchema</code> method will be thrown
     * the same {@link Throwable} object.
     *
     * <p>
     * {@link SchemaFactory} is not allowed to
     * throw {@link SAXException} without first reporting it to
     * {@link ErrorHandler}.
     *
     * <p>
     * Applications can call this method even during a {@link Schema}
     * is being parsed.
     *
     * <p>
     * When the {@link ErrorHandler} is null, the implementation will
     * behave as if the following {@link ErrorHandler} is set:
     * <pre>
     * class DraconianErrorHandler implements {@link ErrorHandler} {
     *     public void fatalError( {@link org.xml.sax.SAXParseException} e ) throws {@link
         SAXException} {
```

```
     *         throw e;
     *     }
     *     public void error( {@link org.xml.sax.SAXParseException} e ) throws {@link SAXException}
          {
     *         throw e;
     *     }
     *     public void warning( {@link org.xml.sax.SAXParseException} e ) throws {@link
         SAXException} {
     *         // noop
     *     }
     * }
     * </pre>
     *
     * <p>
     * When a new {@link SchemaFactory} object is created, initially
     * this field is set to null. This field will <em>NOT</em> be
     * inherited to {@link Schema}s, {@link Validator}s, or
     * {@link ValidatorHandler}s that are created from this {@link SchemaFactory}.
     *
     * @param errorHandler A new error handler to be set.
     *   This parameter can be <code>null</code>.
     */
    public abstract void setErrorHandler(ErrorHandler errorHandler);

    /**
     * Gets the current {@link ErrorHandler} set to this {@link SchemaFactory}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setErrorHandler(ErrorHandler)} method, or null
     *      if that method has never been called since this {@link SchemaFactory}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract ErrorHandler getErrorHandler();

    /**
     * Sets the {@link LSResourceResolver} to customize
     * resource resolution when parsing schemas.
     *
     * <p>
     * {@link SchemaFactory} uses a {@link LSResourceResolver}
     * when it needs to locate external resources while parsing schemas,
     * although exactly what constitutes "locating external resources" is
     * up to each schema language. For example, for W3C XML Schema,
     * this includes files <code>&lt;include></code>d or <code>&lt;import></code>ed,
     * and DTD referenced from schema files, etc.
     *
     * <p>
     * Applications can call this method even during a {@link Schema}
     * is being parsed.
     *
```

```
     * <p>
     * When the {@link LSResourceResolver} is null, the implementation will
     * behave as if the following {@link LSResourceResolver} is set:
     * <pre>
     * class DumbDOMResourceResolver implements {@link LSResourceResolver} {
     *     public {@link org.w3c.dom.ls.LSInput} resolveResource(
     *         String publicId, String systemId, String baseURI) {
     *
     *         return null; // always return null
     *     }
     * }
     * </pre>
     *
     * <p>
     * If a {@link LSResourceResolver} throws a {@link RuntimeException}
     *  (or instances of its derived classes),
     * then the {@link SchemaFactory} will abort the parsing and
     * the caller of the <code>newSchema</code> method will receive
     * the same {@link RuntimeException}.
     *
     * <p>
     * When a new {@link SchemaFactory} object is created, initially
     * this field is set to null.  This field will <em>NOT</em> be
     * inherited to {@link Schema}s, {@link Validator}s, or
     * {@link ValidatorHandler}s that are created from this {@link SchemaFactory}.
     *
     * @param   resourceResolver
     *      A new resource resolver to be set. This parameter can be null.
     */
    public abstract void setResourceResolver(LSResourceResolver resourceResolver);

    /**
     * Gets the current {@link LSResourceResolver} set to this {@link SchemaFactory}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setResourceResolver(LSResourceResolver)} method, or null
     *      if that method has never been called since this {@link SchemaFactory}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract LSResourceResolver getResourceResolver();

    /**
     * <p>Parses the specified source as a schema and returns it as a schema.</p>
     *
     * <p>This is a convenience method for {@link #newSchema(Source[] schemas)}.</p>
     *
     * @param schema Source that represents a schema.
     *
     * @return New <code>Schema</code> from parsing <code>schema</code>.
     *
```

```
     * @throws SAXException If a SAX error occurs during parsing.
     * @throws NullPointerException if <code>schema</code> is null.
     */
    public Schema newSchema(Source schema) throws SAXException {
        return newSchema(new Source[]{schema});
    }

    /**
     * <p>Parses the specified <code>File</code> as a schema and returns it as a <code>Schema</code>.</p>
     *
     * <p>This is a convenience method for {@link #newSchema(Source schema)}.</p>
     *
     * @param schema File that represents a schema.
     *
     * @return New <code>Schema</code> from parsing <code>schema</code>.
     *
     * @throws SAXException If a SAX error occurs during parsing.
     * @throws NullPointerException if <code>schema</code> is null.
     */
    public Schema newSchema(File schema) throws SAXException {
        return newSchema(new StreamSource(schema));
    }

    /**
     * <p>Parses the specified <code>URL</code> as a schema and returns it as a <code>Schema</code>.</p>
     *
     * <p>This is a convenience method for {@link #newSchema(Source schema)}.</p>
     *
     * @param schema <code>URL</code> that represents a schema.
     *
     * @return New <code>Schema</code> from parsing <code>schema</code>.
     *
     * @throws SAXException If a SAX error occurs during parsing.
     * @throws NullPointerException if <code>schema</code> is null.
     */
    public Schema newSchema(URL schema) throws SAXException {
        return newSchema(new StreamSource(schema.toExternalForm()));
    }

    /**
     * Parses the specified source(s) as a schema and returns it as a schema.
     *
     * <p>
     * The callee will read all the {@link Source}s and combine them into a
     * single schema. The exact semantics of the combination depends on the schema
     * language that this {@link SchemaFactory} object is created for.
     *
     * <p>
     * When an {@link ErrorHandler} is set, the callee will report all the errors
     * found in sources to the handler. If the handler throws an exception, it will
     * abort the schema compilation and the same exception will be thrown from
```

```
     * this method. Also, after an error is reported to a handler, the callee is allowed
     * to abort the further processing by throwing it. If an error handler is not set,
     * the callee will throw the first error it finds in the sources.
     *
     * <h2>W3C XML Schema 1.0</h2>
     * <p>
     * The resulting schema contains components from the specified sources.
     * The same result would be achieved if all these sources were
     * imported, using appropriate values for schemaLocation and namespace,
     * into a single schema document with a different targetNamespace
     * and no components of its own, if the import elements were given
     * in the same order as the sources.  Section 4.2.3 of the XML Schema
     * recommendation describes the options processors have in this
     * regard.  While a processor should be consistent in its treatment of
     * JAXP schema sources and XML Schema imports, the behaviour between
     * JAXP-compliant parsers may vary; in particular, parsers may choose
     * to ignore all but the first &lt;import> for a given namespace,
     * regardless of information provided in schemaLocation.
     *
     * <p>
     * If the parsed set of schemas includes error(s) as
     * specified in the section 5.1 of the XML Schema spec, then
     * the error must be reported to the {@link ErrorHandler}.
     *
     * <h2>RELAX NG</h2>
     *
     * <p>For RELAX NG, this method must throw {@link UnsupportedOperationException}
     * if <code>schemas.length!=1</code>.
     *
     *
     * @param schemas
     *      inputs to be parsed. {@link SchemaFactory} is required
     *      to recognize {@link javax.xml.transform.sax.SAXSource},
     *      {@link StreamSource},
     *      {@link javax.xml.transform.stax.StAXSource},
     *      and {@link javax.xml.transform.dom.DOMSource}.
     *      Input schemas must be XML documents or
     *      XML elements and must not be null. For backwards compatibility,
     *      the results of passing anything other than
     *      a document or element are implementation-dependent.
     *      Implementations must either recognize and process the input
     *      or thrown an IllegalArgumentException.
     *
     * @return
     *      Always return a non-null valid {@link Schema} object.
     *      Note that when an error has been reported, there is no
     *      guarantee that the returned {@link Schema} object is
     *      meaningful.
     *
     * @throws SAXException
     *      If an error is found during processing the specified inputs.
     *      When an {@link ErrorHandler} is set, errors are reported to
     *      there first. See {@link #setErrorHandler(ErrorHandler)}.
```

```
     * @throws NullPointerException
     *      If the <code>schemas</code> parameter itself is null or
     *      any item in the array is null.
     * @throws IllegalArgumentException
     *      If any item in the array is not recognized by this method.
     * @throws UnsupportedOperationException
     *      If the schema language doesn't support this operation.
     */
    public abstract Schema newSchema(Source[] schemas) throws SAXException;

    /**
     * Creates a special {@link Schema} object.
     *
     * <p>The exact semantics of the returned {@link Schema} object
     * depend on the schema language for which this {@link SchemaFactory}
     * is created.
     *
     * <p>Also, implementations are allowed to use implementation-specific
     * property/feature to alter the semantics of this method.</p>
     *
     * <p>Implementors and developers should pay particular attention
     * to how the features set on this {@link SchemaFactory} are
     * processed by this special {@link Schema}.
     * In some cases, for example, when the
     * {@link SchemaFactory} and the class actually loading the
     * schema come from different implementations, it may not be possible
     * for {@link SchemaFactory} features to be inherited automatically.
     * Developers should
     * make sure that features, such as secure processing, are explicitly
     * set in both places.</p>
     *
     * <h2>W3C XML Schema 1.0</h2>
     * <p>
     * For XML Schema, this method creates a {@link Schema} object that
     * performs validation by using location hints specified in documents.
     *
     * <p>
     * The returned {@link Schema} object assumes that if documents
     * refer to the same URL in the schema location hints,
     * they will always resolve to the same schema document. This
     * asusmption allows implementations to reuse parsed results of
     * schema documents so that multiple validations against the same
     * schema will run faster.
     *
     * <p>
     * Note that the use of schema location hints introduces a
     * vulnerability to denial-of-service attacks.
     *
     *
     * <h2>RELAX NG</h2>
     * <p>
     * RELAX NG does not support this operation.
     *
```

```
     * @return
     *      Always return non-null valid {@link Schema} object.
     *
     * @throws UnsupportedOperationException
     *      If this operation is not supported by the callee.
     * @throws SAXException
     *      If this operation is supported but failed for some reason.
     */
    public abstract Schema newSchema() throws SAXException;
}
```

26.3 SchemaFactoryConfigurationError.java

Secuencia 253: javax/xml/validation/SchemaFactoryConfigurationError.java

```
/*
 * Copyright (c) 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

/**
 * Thrown when a problem with configuration with the Schema Factories
 * exists. This error will typically be thrown when the class of a
 * schema factory specified in the system properties cannot be found
 * or instantiated.
 * @since 1.8
 */
public final class SchemaFactoryConfigurationError extends Error {

    static final long serialVersionUID = 3531438703147750126L;

    /**
```

```
     * Create a new <code>SchemaFactoryConfigurationError</code> with no
     * detail message.
     */
    public SchemaFactoryConfigurationError() {
    }


    /**
     * Create a new <code>SchemaFactoryConfigurationError</code> with
     * the <code>String</code> specified as an error message.
     *
     * @param message The error message for the exception.
     */
    public SchemaFactoryConfigurationError(String message) {
        super(message);
    }

    /**
     * Create a new <code>SchemaFactoryConfigurationError</code> with the
     * given <code>Throwable</code> base cause.
     *
     * @param cause The exception or error to be encapsulated in a
     * SchemaFactoryConfigurationError.
     */
    public SchemaFactoryConfigurationError(Throwable cause) {
        super(cause);
    }

    /**
     * Create a new <code>SchemaFactoryConfigurationError</code> with the
     * given <code>Throwable</code> base cause and detail message.
     *
     * @param cause The exception or error to be encapsulated in a
     * SchemaFactoryConfigurationError.
     * @param message The detail message.
     */
    public SchemaFactoryConfigurationError(String message, Throwable cause) {
        super(message, cause);
    }

}
```

26.4 SchemaFactoryFinder.java

Secuencia 254: javax/xml/validation/SchemaFactoryFinder.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import java.io.File;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.net.URL;
import java.security.AccessControlContext;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Properties;
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * Implementation of {@link SchemaFactory#newInstance(String)}.
 *
 * @author <a href="Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @version $Revision: 1.8 $, $Date: 2010-11-01 04:36:13 $
 * @since 1.5
 */
class SchemaFactoryFinder  {

    /** debug support code. */
    private static boolean debug = false;
    /**
     *<p> Take care of restrictions imposed by java security model </p>
     */
    private static final SecuritySupport ss = new SecuritySupport();
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xerces.internal";
    /**
     * <p>Cache properties for performance.</p>
     */
    private static final Properties cacheProps = new Properties();

    /**
     * <p>First time requires initialization overhead.</p>
```

```
     */
    private static volatile boolean firstTime = true;

    static {
        // Use try/catch block to support applets
        try {
            debug = ss.getSystemProperty("jaxp.debug") != null;
        } catch (Exception unused) {
            debug = false;
        }
    }

    /**
     * <p>Conditional debug printing.</p>
     *
     * @param msg to print
     */
    private static void debugPrintln(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * <p><code>ClassLoader</code> to use to find <code>SchemaFactory</code>.</p>
     */
    private final ClassLoader classLoader;

    /**
     * <p>Constructor that specifies <code>ClassLoader</code> to use
     * to find <code>SchemaFactory</code>.</p>
     *
     * @param loader
     *      to be used to load resource, {@link SchemaFactory}, and
     *      {@link SchemaFactoryLoader} implementations during
     *      the resolution process.
     *      If this parameter is null, the default system class loader
     *      will be used.
     */
    public SchemaFactoryFinder(ClassLoader loader) {
        this.classLoader = loader;
        if( debug ) {
            debugDisplayClassLoader();
        }
    }

    private void debugDisplayClassLoader() {
        try {
            if( classLoader == ss.getContextClassLoader() ) {
                debugPrintln("using thread context class loader ("+classLoader+") for search");
                return;
            }
        } catch( Throwable unused ) {
```

```
            // getContextClassLoader() undefined in JDK1.1
        }

        if( classLoader==ClassLoader.getSystemClassLoader() ) {
            debugPrintln("using system class loader ("+classLoader+") for search");
            return;
        }

        debugPrintln("using class loader ("+classLoader+") for search");
    }

    /**
     * <p>Creates a new {@link SchemaFactory} object for the specified
     * schema language.</p>
     *
     * @param schemaLanguage
     *      See {@link SchemaFactory Schema Language} table in <code>SchemaFactory</code>
     *      for the list of available schema languages.
     *
     * @return <code>null</code> if the callee fails to create one.
     *
     * @throws NullPointerException
     *      If the <code>schemaLanguage</code> parameter is null.
     * @throws SchemaFactoryConfigurationError
     *      If a configuration error is encountered.
     */
    public SchemaFactory newFactory(String schemaLanguage) {
        if(schemaLanguage==null) {
            throw new NullPointerException();
        }
        SchemaFactory f = _newFactory(schemaLanguage);
        if (f != null) {
            debugPrintln("factory '" + f.getClass().getName() + "' was found for " + schemaLanguage
                );
        } else {
            debugPrintln("unable to find a factory for " + schemaLanguage);
        }
        return f;
    }

    /**
     * <p>Lookup a <code>SchemaFactory</code> for the given <code>schemaLanguage</code>.</p>
     *
     * @param schemaLanguage Schema language to lookup <code>SchemaFactory</code> for.
     *
     * @return <code>SchemaFactory</code> for the given <code>schemaLanguage</code>.
     */
    private SchemaFactory _newFactory(String schemaLanguage) {
        SchemaFactory sf;

        String propertyName = SERVICE_CLASS.getName() + ":" + schemaLanguage;

        // system property look up
```

```
        try {
            debugPrintln("Looking up system property '"+propertyName+"'" );
            String r = ss.getSystemProperty(propertyName);
            if(r!=null) {
                debugPrintln("The value is '"+r+"'");
                sf = createInstance(r, true);
                if(sf!=null)    return sf;
            } else
                debugPrintln("The property is undefined.");
        } catch( Throwable t ) {
            if( debug ) {
                debugPrintln("failed to look up system property '"+propertyName+"'" );
                t.printStackTrace();
            }
        }

        String javah = ss.getSystemProperty( "java.home" );
        String configFile = javah + File.separator +
        "lib" + File.separator + "jaxp.properties";


        // try to read from $java.home/lib/jaxp.properties
        try {
            if(firstTime){
                synchronized(cacheProps){
                    if(firstTime){
                        File f=new File( configFile );
                        firstTime = false;
                        if(ss.doesFileExist(f)){
                            debugPrintln("Read properties file " + f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                    }
                }
            }
            final String factoryClassName = cacheProps.getProperty(propertyName);
            debugPrintln("found " + factoryClassName + " in $java.home/jaxp.properties");

            if (factoryClassName != null) {
                sf = createInstance(factoryClassName, true);
                if(sf != null){
                    return sf;
                }
            }
        } catch (Exception ex) {
            if (debug) {
                ex.printStackTrace();
            }
        }

        // Try with ServiceLoader
        final SchemaFactory factoryImpl = findServiceProvider(schemaLanguage);

```

```
        // The following assertion should always be true.
        // Uncomment it, recompile, and run with -ea in case of doubts:
        // assert factoryImpl == null || factoryImpl.isSchemaLanguageSupported(schemaLanguage);

        if (factoryImpl != null) {
            return factoryImpl;
        }

        // platform default
        if(schemaLanguage.equals("http://www.w3.org/2001/XMLSchema")) {
            debugPrintln("attempting to use the platform default XML Schema validator");
            return createInstance("com.sun.org.apache.xerces.internal.jaxp.validation.
                XMLSchemaFactory", true);
        }

        debugPrintln("all things were tried, but none was found. bailing out.");
        return null;
    }

    /** <p>Create class using appropriate ClassLoader.</p>
     *
     * @param className Name of class to create.
     * @return Created class or <code>null</code>.
     */
    private Class<?> createClass(String className) {
        Class<?> clazz;
        // make sure we have access to restricted packages
        boolean internal = false;
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                internal = true;
            }
        }

        try {
            if (classLoader != null && !internal) {
                clazz = Class.forName(className, false, classLoader);
            } else {
                clazz = Class.forName(className);
            }
        } catch (Throwable t) {
            if(debug)  {
                t.printStackTrace();
            }
            return null;
        }

        return clazz;
    }

    /**
     * <p>Creates an instance of the specified and returns it.</p>
     *
```

```
     * @param className
     *      fully qualified class name to be instantiated.
     *
     * @return null
     *      if it fails. Error messages will be printed by this method.
     */
    SchemaFactory createInstance( String className ) {
        return createInstance( className, false );
    }

    SchemaFactory createInstance( String className, boolean useServicesMechanism ) {
        SchemaFactory schemaFactory = null;

        debugPrintln("createInstance(" + className + ")");

        // get Class from className
        Class<?> clazz = createClass(className);
        if (clazz == null) {
                debugPrintln("failed to getClass(" + className + ")");
                return null;
        }
        debugPrintln("loaded " + className + " from " + which(clazz));

        // instantiate Class as a SchemaFactory
        try {
                if (!SchemaFactory.class.isAssignableFrom(clazz)) {
                    throw new ClassCastException(clazz.getName()
                                + " cannot be cast to " + SchemaFactory.class);
                }
                if (!useServicesMechanism) {
                    schemaFactory = newInstanceNoServiceLoader(clazz);
                }
                if (schemaFactory == null) {
                    schemaFactory = (SchemaFactory) clazz.newInstance();
                }
        } catch (ClassCastException classCastException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        classCastException.printStackTrace();
                }
                return null;
        } catch (IllegalAccessException illegalAccessException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        illegalAccessException.printStackTrace();
                }
                return null;
        } catch (InstantiationException instantiationException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        instantiationException.printStackTrace();
                }
                return null;
```

```java
        }

        return schemaFactory;
    }

    /**
     * Try to construct using newXMLSchemaFactoryNoServiceLoader
     *   method if available.
     */
    private static SchemaFactory newInstanceNoServiceLoader(
         Class<?> providerClass
    ) {
        // Retain maximum compatibility if no security manager.
        if (System.getSecurityManager() == null) {
            return null;
        }
        try {
            final Method creationMethod =
                providerClass.getDeclaredMethod(
                    "newXMLSchemaFactoryNoServiceLoader"
                );
            final int modifiers = creationMethod.getModifiers();

            // Do not call the method if it's not public static.
            if (!Modifier.isStatic(modifiers) || !Modifier.isPublic(modifiers)) {
                return null;
            }

            // Only calls "newXMLSchemaFactoryNoServiceLoader" if it's
            // declared to return an instance of SchemaFactory.
            final Class<?> returnType = creationMethod.getReturnType();
            if (SERVICE_CLASS.isAssignableFrom(returnType)) {
                return SERVICE_CLASS.cast(creationMethod.invoke(null, (Object[])null));
            } else {
                // Should not happen since
                // XMLSchemaFactory.newXMLSchemaFactoryNoServiceLoader is
                // declared to return XMLSchemaFactory.
                throw new ClassCastException(returnType
                            + " cannot be cast to " + SERVICE_CLASS);
            }
        } catch(ClassCastException e) {
            throw new SchemaFactoryConfigurationError(e.getMessage(), e);
        } catch (NoSuchMethodException exc) {
            return null;
        } catch (Exception exc) {
            return null;
        }
    }

    // Call isSchemaLanguageSupported with initial context.
    private boolean isSchemaLanguageSupportedBy(final SchemaFactory factory,
            final String schemaLanguage,
            AccessControlContext acc) {
```

```java
        return AccessController.doPrivileged(new PrivilegedAction<Boolean>() {
            public Boolean run() {
                return factory.isSchemaLanguageSupported(schemaLanguage);
            }
        }, acc);
    }

    /**
     * Finds a service provider subclass of SchemaFactory that supports the
     * given schema language using the ServiceLoader.
     *
     * @param schemaLanguage The schema language for which we seek a factory.
     * @return A SchemaFactory supporting the specified schema language, or null
     *         if none is found.
     * @throws SchemaFactoryConfigurationError if a configuration error is found.
     */
    private SchemaFactory findServiceProvider(final String schemaLanguage) {
        assert schemaLanguage != null;
        // store current context.
        final AccessControlContext acc = AccessController.getContext();
        try {
            return AccessController.doPrivileged(new PrivilegedAction<SchemaFactory>() {
                public SchemaFactory run() {
                    final ServiceLoader<SchemaFactory> loader =
                            ServiceLoader.load(SERVICE_CLASS);
                    for (SchemaFactory factory : loader) {
                        // restore initial context to call
                        // factory.isSchemaLanguageSupported
                        if (isSchemaLanguageSupportedBy(factory, schemaLanguage, acc)) {
                            return factory;
                        }
                    }
                    return null; // no factory found.
                }
            });
        } catch (ServiceConfigurationError error) {
            throw new SchemaFactoryConfigurationError(
                    "Provider for " + SERVICE_CLASS + " cannot be created", error);
        }
    }

    private static final Class<SchemaFactory> SERVICE_CLASS = SchemaFactory.class;


    private static String which( Class<?> clazz ) {
        return which( clazz.getName(), clazz.getClassLoader() );
    }

    /**
     * <p>Search the specified classloader for the given classname.</p>
     *
     * @param classname the fully qualified name of the class to search for
     * @param loader the classloader to search
```

```
     *
     * @return the source location of the resource, or null if it wasn't found
     */
    private static String which(String classname, ClassLoader loader) {

        String classnameAsResource = classname.replace('.', '/') + ".class";

        if( loader==null )  loader = ClassLoader.getSystemClassLoader();

        //URL it = loader.getResource(classnameAsResource);
        URL it = ss.getResourceAsURL(loader, classnameAsResource);
        if (it != null) {
            return it.toString();
        } else {
            return null;
        }
    }
}
```

26.5 SchemaFactoryLoader.java

Secuencia 255: javax/xml/validation/SchemaFactoryLoader.java

```
/*
 * Copyright (c) 2004, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

/**
 * <p>Factory that creates {@link SchemaFactory}.</p>
 *
 * <p><b>DO NOT USE THIS CLASS</b></p>
```

```
 *
 * <p>
 * This class was introduced as a part of an early proposal during the
 * JSR-206 standardization process. The proposal was eventually abandoned
 * but this class accidentally remained in the source tree, and made its
 * way into the final version.
 * </p><p>
 * This class does not participate in any JAXP 1.3 or JAXP 1.4 processing.
 * It must not be used by users or JAXP implementations.
 * </p>
 *
 * @author <a href="Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @since 1.5
 */
public abstract class SchemaFactoryLoader {

    /**
     * A do-nothing constructor.
     */
    protected SchemaFactoryLoader() {
    }

    /**
     * Creates a new {@link SchemaFactory} object for the specified
     * schema language.
     *
     * @param schemaLanguage
     *      See <a href="SchemaFactory.html#schemaLanguage">
     *      the list of available schema languages</a>.
     *
     * @throws NullPointerException
     *      If the <tt>schemaLanguage</tt> parameter is null.
     *
     * @return <code>null</code> if the callee fails to create one.
     */
    public abstract SchemaFactory newFactory(String schemaLanguage);
}
```

26.6 SecuritySupport.java

Secuencia 256: javax/xml/validation/SecuritySupport.java

```
/*
 * Copyright (c) 2005, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import java.io.IOException;
import java.net.URL;
import java.security.*;
import java.net.*;
import java.io.*;
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


    ClassLoader getContextClassLoader() {
        return (ClassLoader)
        AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                //try {
                cl = Thread.currentThread().getContextClassLoader();
                //} catch (SecurityException ex) { }
                if (cl == null)
                    cl = ClassLoader.getSystemClassLoader();
                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
```

```
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getURLInputStream(final URL url)
        throws IOException
    {
        try {
            return (InputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws IOException {
                        return url.openStream();
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (IOException)e.getException();
        }
    }

    URL getResourceAsURL(final ClassLoader cl,
                                        final String name)
    {
        return (URL)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    URL url;
                    if (cl == null) {
                        url = Object.class.getResource(name);
                    } else {
                        url = cl.getResource(name);
                    }
                    return url;
                }
            });
    }

    Enumeration getResources(final ClassLoader cl,
                                        final String name) throws IOException
    {
```

```
        try{
        return (Enumeration)
            AccessController.doPrivileged(new PrivilegedExceptionAction() {
                public Object run() throws IOException{
                    Enumeration enumeration;
                    if (cl == null) {
                        enumeration = ClassLoader.getSystemResources(name);
                    } else {
                        enumeration = cl.getResources(name);
                    }
                    return enumeration;
                }
            });
        }catch(PrivilegedActionException e){
            throw (IOException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                        final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

}
```

26.7 TypeInfoProvider.java

Secuencia 257: javax/xml/validation/TypeInfoProvider.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import org.w3c.dom.TypeInfo;

/**
 * This class provides access to the type information determined
 * by {@link ValidatorHandler}.
 *
 * <p>
 * Some schema languages, such as W3C XML Schema, encourages a validator
 * to report the "type" it assigns to each attribute/element.
 * Those applications who wish to access this type information can invoke
 * methods defined on this "interface" to access such type information.
 *
 * <p>
 * Implementation of this "interface" can be obtained through the
 * {@link ValidatorHandler#getTypeInfoProvider()} method.
 *
 * @author  <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @see org.w3c.dom.TypeInfo
 * @since 1.5
 */
public abstract class TypeInfoProvider {

    /**
     * Constructor for the derived class.
     *
     * <p>
     * The constructor does nothing.
     */
    protected TypeInfoProvider() {
    }
```

```

    /**
     * <p>Returns the immutable {@link TypeInfo} object for the current
     * element.</p>
     *
     * <p>The method may only be called by the startElement event
     * or the endElement event
     * of the {@link org.xml.sax.ContentHandler} that the application sets to
     * the {@link ValidatorHandler}.</p>
     *
     * <p>When W3C XML Schema validation is being performed, in the
     * case where an element has a union type, the {@link TypeInfo}
     * returned by a call to <code>getElementTypeInfo()</code> from the
     * startElement
     * event will be the union type. The <code>TypeInfo</code>
     * returned by a call
     * from the endElement event will be the actual member type used
     * to validate the element.</p>
     *
     * @throws IllegalStateException
     *      If this method is called from other {@link org.xml.sax.ContentHandler}
     *      methods.
     * @return
     *      An immutable {@link TypeInfo} object that represents the
     *      type of the current element.
     *      Note that the caller can keep references to the obtained
     *      {@link TypeInfo} longer than the callback scope.
     *
     *      Otherwise, this method returns
     *      null if the validator is unable to
     *      determine the type of the current element for some reason
     *      (for example, if the validator is recovering from
     *      an earlier error.)
     *
     */
    public abstract TypeInfo getElementTypeInfo();

    /**
     * Returns the immutable {@link TypeInfo} object for the specified
     * attribute of the current element.
     *
     * <p>
     * The method may only be called by the startElement event of
     * the {@link org.xml.sax.ContentHandler} that the application sets to the
     * {@link ValidatorHandler}.</p>
     *
     * @param index
     *      The index of the attribute. The same index for
     *      the {@link org.xml.sax.Attributes} object passed to the
     *      <code>startElement</code> callback.
     *
     * @throws IndexOutOfBoundsException
     *      If the index is invalid.
```

```
     * @throws IllegalStateException
     *      If this method is called from other {@link org.xml.sax.ContentHandler}
     *      methods.
     *
     * @return
     *      An immutable {@link TypeInfo} object that represents the
     *      type of the specified attribute.
     *      Note that the caller can keep references to the obtained
     *      {@link TypeInfo} longer than the callback scope.
     *
     *      Otherwise, this method returns
     *      null if the validator is unable to
     *      determine the type.
     */
    public abstract TypeInfo getAttributeTypeInfo(int index);

    /**
     * Returns <code>true</code> if the specified attribute is determined
     * to be ID.
     *
     * <p>
     * Exacly how an attribute is "determined to be ID" is up to the
     * schema language. In case of W3C XML Schema, this means
     * that the actual type of the attribute is the built-in ID type
     * or its derived type.
     *
     * <p>
     * A {@link javax.xml.parsers.DocumentBuilder} uses this information
     * to properly implement {@link org.w3c.dom.Attr#isId()}.
     *
     * <p>
     * The method may only be called by the startElement event of
     * the {@link org.xml.sax.ContentHandler} that the application sets to the
     * {@link ValidatorHandler}.
     *
     * @param index
     *      The index of the attribute. The same index for
     *      the {@link org.xml.sax.Attributes} object passed to the
     *      <code>startElement</code> callback.
     *
     * @throws IndexOutOfBoundsException
     *      If the index is invalid.
     * @throws IllegalStateException
     *      If this method is called from other {@link org.xml.sax.ContentHandler}
     *      methods.
     *
     * @return true
     *      if the type of the specified attribute is ID.
     */
    public abstract boolean isIdAttribute(int index);

    /**
     * Returns <code>false</code> if the attribute was added by the validator.
```

```
     *
     * <p>
     * This method provides information necessary for
     * a {@link javax.xml.parsers.DocumentBuilder} to determine what
     * the DOM tree should return from the {@link org.w3c.dom.Attr#getSpecified()} method.
     *
     * <p>
     * The method may only be called by the startElement event of
     * the {@link org.xml.sax.ContentHandler} that the application sets to the
     * {@link ValidatorHandler}.
     *
     * <p>
     * A general guideline for validators is to return true if
     * the attribute was originally present in the pipeline, and
     * false if it was added by the validator.
     *
     * @param index
     *      The index of the attribute. The same index for
     *      the {@link org.xml.sax.Attributes} object passed to the
     *      <code>startElement</code> callback.
     *
     * @throws IndexOutOfBoundsException
     *      If the index is invalid.
     * @throws IllegalStateException
     *      If this method is called from other {@link org.xml.sax.ContentHandler}
     *      methods.
     *
     * @return
     *      <code>true</code> if the attribute was present before the validator
     *      processes input. <code>false</code> if the attribute was added
     *      by the validator.
     */
    public abstract boolean isSpecified(int index);
}
```

26.8 Validator.java

Secuencia 258: javax/xml/validation/Validator.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import java.io.IOException;

import javax.xml.transform.Result;
import javax.xml.transform.Source;

import org.w3c.dom.ls.LSResourceResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;

/**
 * <p>A processor that checks an XML document against {@link Schema}.</p>
 *
 * <p>
 * A validator object is not thread-safe and not reentrant.
 * In other words, it is the application's responsibility to make
 * sure that one {@link Validator} object is not used from
 * more than one thread at any given time, and while the <code>validate</code>
 * method is invoked, applications may not recursively call
 * the <code>validate</code> method.
 * <p>
 *
 *
 * @author  <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @since 1.5
 */
public abstract class Validator {

    /**
     * Constructor for derived classes.
     *
     * <p>The constructor does nothing.</p>
     *
     * <p>Derived classes must create {@link Validator} objects that have
     * <code>null</code> {@link ErrorHandler} and
     * <code>null</code> {@link LSResourceResolver}.
     * </p>
     */
    protected Validator() {
    }
```

```

        /**
         * <p>Reset this <code>Validator</code> to its original configuration.</p>
         *
         * <p><code>Validator</code> is reset to the same state as when it was created with
         * {@link Schema#newValidator()}.
         * <code>reset()</code> is designed to allow the reuse of existing <code>Validator</code>s
         * thus saving resources associated with the creation of new <code>Validator</code>s.</p>
         *
         * <p>The reset <code>Validator</code> is not guaranteed to have the same {@link
             LSResourceResolver} or {@link ErrorHandler}
         * <code>Object</code>s, e.g. {@link Object#equals(Object obj)}.  It is guaranteed to have
             a functionally equal
         * <code>LSResourceResolver</code> and <code>ErrorHandler</code>.</p>
         */
        public abstract void reset();

    /**
     * Validates the specified input.
     *
     * <p>This is just a convenience method for
     * {@link #validate(Source source, Result result)}
     * with <code>result</code> of <code>null</code>.</p>
     *
     * @param source
     *      XML to be validated. Must be an XML document or
     *      XML element and must not be null. For backwards compatibility,
     *      the results of attempting to validate anything other than
     *      a document or element are implementation-dependent.
     *      Implementations must either recognize and process the input
     *      or throw an IllegalArgumentException.
     *
     * @throws IllegalArgumentException
     *      If the <code>Source</code>
     *      is an XML artifact that the implementation cannot
     *      validate (for example, a processing instruction).
     *
     * @throws SAXException
     *      If the {@link ErrorHandler} throws a {@link SAXException} or
     *      if a fatal error is found and the {@link ErrorHandler} returns
     *      normally.
     *
     * @throws IOException
     *      If the validator is processing a
     *      {@link javax.xml.transform.sax.SAXSource} and the
     *      underlying {@link org.xml.sax.XMLReader} throws an
     *      {@link IOException}.
     *
     *
     * @throws NullPointerException If <code>source</code> is
     *   <code>null</code>.
     *
     * @see #validate(Source source, Result result)
```

```
     */
    public void validate(Source source)
        throws SAXException, IOException {

        validate(source, null);
    }

    /**
     * <p>Validates the specified input and send the augmented validation
     * result to the specified output.</p>
     *
     * <p>This method places the following restrictions on the types of
     * the {@link Source}/{@link Result} accepted.</p>
     *
     * <table border=1>
     * <thead>
     *  <tr>
     *   <th colspan="5"><code>Source</code> / <code>Result</code> Accepted</th>
     *  </tr>
     *  <tr>
     *   <th></th>
     *   <th>{@link javax.xml.transform.stream.StreamSource}</th>
     *   <th>{@link javax.xml.transform.sax.SAXSource}</th>
     *   <th>{@link javax.xml.transform.dom.DOMSource}</th>
     *   <th>{@link javax.xml.transform.stax.StAXSource}</th>
     *  </tr>
     * </thead>
     * <tbody align="center">
     *  <tr>
     *   <td><code>null</code></td>
     *   <td>OK</td>
     *   <td>OK</td>
     *   <td>OK</td>
     *   <td>OK</td>
     *  </tr>
     *  <tr>
     *   <th>{@link javax.xml.transform.stream.StreamResult}</th>
     *   <td>OK</td>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
     *  </tr>
     *  <tr>
     *   <th>{@link javax.xml.transform.sax.SAXResult}</th>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td>OK</td>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
     *  </tr>
     *  <tr>
     *   <th>{@link javax.xml.transform.dom.DOMResult}</th>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
```

```
     *   <td>OK</td>
     *   <td><code>IllegalArgumentException</code></td>
     *  </tr>
     *  <tr>
     *   <th>{@link javax.xml.transform.stax.StAXResult}</th>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td><code>IllegalArgumentException</code></td>
     *   <td>OK</td>
     *  </tr>
     * </tbody>
     * </table>
     *
     * <p>To validate one <code>Source</code> into another kind of
     * <code>Result</code>, use the identity transformer (see
     * {@link javax.xml.transform.TransformerFactory#newTransformer()}).</p>
     *
     * <p>Errors found during the validation is sent to the specified
     * {@link ErrorHandler}.</p>
     *
     * <p>If a document is valid, or if a document contains some errors
     * but none of them were fatal and the <code>ErrorHandler</code> didn't
     * throw any exception, then the method returns normally.</p>
     *
     * @param source
     *      XML to be validated. Must be an XML document or
     *      XML element and must not be null. For backwards compatibility,
     *      the results of attempting to validate anything other than
     *      a document or element are implementation-dependent.
     *      Implementations must either recognize and process the input
     *      or throw an IllegalArgumentException.
     *
     * @param result
     *      The <code>Result</code> object that receives (possibly augmented)
     *      XML. This parameter can be null if the caller is not interested
     *      in it.
     *
     *      Note that when a <code>DOMResult</code> is used,
     *      a validator might just pass the same DOM node from
     *      <code>DOMSource</code> to <code>DOMResult</code>
     *      (in which case <code>source.getNode()==result.getNode()</code>),
     *      it might copy the entire DOM tree, or it might alter the
     *      node given by the source.
     *
     * @throws IllegalArgumentException
     *      If the <code>Result</code> type doesn't match the
     *      <code>Source</code> type of if the <code>Source</code>
     *      is an XML artifact that the implementation cannot
     *      validate (for example, a processing instruction).
     * @throws SAXException
     *      If the <code>ErrorHandler</code> throws a
     *      <code>SAXException</code> or
     *      if a fatal error is found and the <code>ErrorHandler</code> returns
```

```
     *      normally.
     * @throws IOException
     *      If the validator is processing a
     *      <code>SAXSource</code> and the
     *      underlying {@link org.xml.sax.XMLReader} throws an
     *      <code>IOException</code>.
     * @throws NullPointerException
     *      If the <code>source</code> parameter is <code>null</code>.
     *
     * @see #validate(Source source)
     */
    public abstract void validate(Source source, Result result)
        throws SAXException, IOException;

    /**
     * Sets the {@link ErrorHandler} to receive errors encountered
     * during the <code>validate</code> method invocation.
     *
     * <p>
     * Error handler can be used to customize the error handling process
     * during a validation. When an {@link ErrorHandler} is set,
     * errors found during the validation will be first sent
     * to the {@link ErrorHandler}.
     *
     * <p>
     * The error handler can abort further validation immediately
     * by throwing {@link SAXException} from the handler. Or for example
     * it can print an error to the screen and try to continue the
     * validation by returning normally from the {@link ErrorHandler}
     *
     * <p>
     * If any {@link Throwable} is thrown from an {@link ErrorHandler},
     * the caller of the <code>validate</code> method will be thrown
     * the same {@link Throwable} object.
     *
     * <p>
     * {@link Validator} is not allowed to
     * throw {@link SAXException} without first reporting it to
     * {@link ErrorHandler}.
     *
     * <p>
     * When the {@link ErrorHandler} is null, the implementation will
     * behave as if the following {@link ErrorHandler} is set:
     * <pre>
     * class DraconianErrorHandler implements {@link ErrorHandler} {
     *     public void fatalError( {@link org.xml.sax.SAXParseException} e ) throws {@link
         SAXException} {
     *         throw e;
     *     }
     *     public void error( {@link org.xml.sax.SAXParseException} e ) throws {@link SAXException}
         {
     *         throw e;
     *     }
```

```
     *     public void warning( {@link org.xml.sax.SAXParseException} e ) throws {@link
         SAXException} {
     *         // noop
     *     }
     * }
     * </pre>
     *
     * <p>
     * When a new {@link Validator} object is created, initially
     * this field is set to null.
     *
     * @param   errorHandler
     *      A new error handler to be set. This parameter can be null.
     */
    public abstract void setErrorHandler(ErrorHandler errorHandler);

    /**
     * Gets the current {@link ErrorHandler} set to this {@link Validator}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setErrorHandler(ErrorHandler)} method, or null
     *      if that method has never been called since this {@link Validator}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract ErrorHandler getErrorHandler();

    /**
     * Sets the {@link LSResourceResolver} to customize
     * resource resolution while in a validation episode.
     *
     * <p>
     * {@link Validator} uses a {@link LSResourceResolver}
     * when it needs to locate external resources while a validation,
     * although exactly what constitutes "locating external resources" is
     * up to each schema language.
     *
     * <p>
     * When the {@link LSResourceResolver} is null, the implementation will
     * behave as if the following {@link LSResourceResolver} is set:
     * <pre>
     * class DumbLSResourceResolver implements {@link LSResourceResolver} {
     *     public {@link org.w3c.dom.ls.LSInput} resolveResource(
     *         String publicId, String systemId, String baseURI) {
     *
     *         return null; // always return null
     *     }
     * }
     * </pre>
     *
     * <p>
```

```
     * If a {@link LSResourceResolver} throws a {@link RuntimeException}
     *  (or instances of its derived classes),
     * then the {@link Validator} will abort the parsing and
     * the caller of the <code>validate</code> method will receive
     * the same {@link RuntimeException}.
     *
     * <p>
     * When a new {@link Validator} object is created, initially
     * this field is set to null.
     *
     * @param   resourceResolver
     *      A new resource resolver to be set. This parameter can be null.
     */
    public abstract void setResourceResolver(LSResourceResolver resourceResolver);

    /**
     * Gets the current {@link LSResourceResolver} set to this {@link Validator}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setResourceResolver(LSResourceResolver)} method, or null
     *      if that method has never been called since this {@link Validator}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract LSResourceResolver getResourceResolver();



    /**
     * Look up the value of a feature flag.
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for a {@link Validator} to recognize a feature name but
     * temporarily be unable to return its value.
     * Some feature values may be available only in specific
     * contexts, such as before, during, or after a validation.
     *
     * <p>Implementors are free (and encouraged) to invent their own features,
     * using names built on their own URIs.</p>
     *
     * @param name The feature name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link Validator} recognizes the feature name but
     *   cannot determine its value at this time.
     * @throws NullPointerException
     *   When the name parameter is null.
```

```
     *
     * @see #setFeature(String, boolean)
     */
    public boolean getFeature(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException("the name parameter is null");
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * Set the value of a feature flag.
     *
     * <p>
     * Feature can be used to control the way a {@link Validator}
     * parses schemas, although {@link Validator}s are not required
     * to recognize any specific feature names.</p>
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for a {@link Validator} to expose a feature value but
     * to be unable to change the current value.
     * Some feature values may be immutable or mutable only
     * in specific contexts, such as before, during, or after
     * a validation.</p>
     *
     * @param name The feature name, which is a non-null fully-qualified URI.
     * @param value The requested value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link Validator} recognizes the feature name but
     *   cannot set the requested value.
     * @throws NullPointerException
     *   When the name parameter is null.
     *
     * @see #getFeature(String)
     */
    public void setFeature(String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException("the name parameter is null");
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * Set the value of a property.
```

```
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link Validator} to recognize a property name but
     * to be unable to change the current value.
     * Some property values may be immutable or mutable only
     * in specific contexts, such as before, during, or after
     * a validation.</p>
     *
     * <p>
     * All implementations that implement JAXP 1.5 or newer are required to
     * support the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD} and
     * {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} properties.
     * </p>
     * <ul>
     *   <li>
     *      <p>Access to external DTDs in source or Schema file is restricted to
     *      the protocols specified by the {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_DTD}
     *      property.  If access is denied during validation due to the restriction
     *      of this property, {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link #validate(Source)} method.</p>
     *
     *      <p>Access to external reference set by the schemaLocation attribute is
     *      restricted to the protocols specified by the
     *      {@link javax.xml.XMLConstants#ACCESS_EXTERNAL_SCHEMA} property.
     *      If access is denied during validation due to the restriction of this property,
     *      {@link org.xml.sax.SAXException} will be thrown by the
     *      {@link #validate(Source)} method.</p>
     *   </li>
     * </ul>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
     * @param object The requested value for the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link Validator} recognizes the property name but
     *   cannot set the requested value.
     * @throws NullPointerException
     *   When the name parameter is null.
     */
    public void setProperty(String name, Object object)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException("the name parameter is null");
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * Look up the value of a property.
```

```
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link Validator} to recognize a property name but
     * temporarily be unable to return its value.
     * Some property values may be available only in specific
     * contexts, such as before, during, or after a validation.</p>
     *
     * <p>{@link Validator}s are not required to recognize any specific
     * property names.</p>
     *
     * <p>Implementors are free (and encouraged) to invent their own properties,
     * using names built on their own URIs.</p>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   XMLReader recognizes the property name but
     *   cannot determine its value at this time.
     * @throws NullPointerException
     *   When the name parameter is null.
     *
     * @see #setProperty(String, Object)
     */
    public Object getProperty(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException("the name parameter is null");
        }

        throw new SAXNotRecognizedException(name);
    }
}
```

26.9 ValidatorHandler.java

Secuencia 259: javax/xml/validation/ValidatorHandler.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.validation;

import org.w3c.dom.ls.LSResourceResolver;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;

/**
 * Streaming validator that works on SAX stream.
 *
 * <p>
 * A {@link ValidatorHandler} object is not thread-safe and not reentrant.
 * In other words, it is the application's responsibility to make
 * sure that one {@link ValidatorHandler} object is not used from
 * more than one thread at any given time.
 *
 * <p>
 * {@link ValidatorHandler} checks if the SAX events follow
 * the set of constraints described in the associated {@link Schema},
 * and additionally it may modify the SAX events (for example
 * by adding default values, etc.)
 *
 * <p>
 * {@link ValidatorHandler} extends from {@link ContentHandler},
 * but it refines the underlying {@link ContentHandler} in
 * the following way:
 * <ol>
 *  <li>startElement/endElement events must receive non-null String
 *      for <code>uri</code>, <code>localName</code>, and <code>qname</code>,
 *      even though SAX allows some of them to be null.
 *      Similarly, the user-specified {@link ContentHandler} will receive non-null
 *      Strings for all three parameters.
 *
 *  <li>Applications must ensure that {@link ValidatorHandler}'s
 *      {@link ContentHandler#startPrefixMapping(String,String)} and
 *      {@link ContentHandler#endPrefixMapping(String)} are invoked
 *      properly. Similarly, the user-specified {@link ContentHandler}
 *      will receive startPrefixMapping/endPrefixMapping events.
 *      If the {@link ValidatorHandler} introduces additional namespace
```

```
 *      bindings, the user-specified {@link ContentHandler} will receive
 *      additional startPrefixMapping/endPrefixMapping events.
 *
 *  <li>{@link org.xml.sax.Attributes} for the
 *      {@link ContentHandler#startElement(String,String,String,Attributes)} method
 *      may or may not include xmlns* attributes.
 * </ol>
 *
 * <p>
 * A {@link ValidatorHandler} is automatically reset every time
 * the startDocument method is invoked.
 *
 * <h2>Recognized Properties and Features</h2>
 * <p>
 * This spec defines the following feature that must be recognized
 * by all {@link ValidatorHandler} implementations.
 *
 * <h3><code>http://xml.org/sax/features/namespace-prefixes</code></h3>
 * <p>
 * This feature controls how a {@link ValidatorHandler} introduces
 * namespace bindings that were not present in the original SAX event
 * stream.
 * When this feature is set to true, it must make
 * sure that the user's {@link ContentHandler} will see
 * the corresponding <code>xmlns*</code> attribute in
 * the {@link org.xml.sax.Attributes} object of the
 * {@link ContentHandler#startElement(String,String,String,Attributes)}
 * callback. Otherwise, <code>xmlns*</code> attributes must not be
 * added to {@link org.xml.sax.Attributes} that's passed to the
 * user-specified {@link ContentHandler}.
 * <p>
 * (Note that regardless of this switch, namespace bindings are
 * always notified to applications through
 * {@link ContentHandler#startPrefixMapping(String,String)} and
 * {@link ContentHandler#endPrefixMapping(String)} methods of the
 * {@link ContentHandler} specified by the user.)
 *
 * <p>
 * Note that this feature does <em>NOT</em> affect the way
 * a {@link ValidatorHandler} receives SAX events. It merely
 * changes the way it augments SAX events.
 *
 * <p>This feature is set to <code>false</code> by default.</p>
 *
 * @author  <a href="mailto:Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @since 1.5
 */
public abstract class ValidatorHandler implements ContentHandler {

    /**
     * <p>Constructor for derived classes.</p>
     *
     * <p>The constructor does nothing.</p>
```

```
     *
     * <p>Derived classes must create {@link ValidatorHandler} objects that have
     * <code>null</code> {@link ErrorHandler} and
     * <code>null</code> {@link LSResourceResolver}.</p>
     */
    protected ValidatorHandler() {
    }

    /**
     * Sets the {@link ContentHandler} which receives
     * the augmented validation result.
     *
     * <p>
     * When a {@link ContentHandler} is specified, a
     * {@link ValidatorHandler} will work as a filter
     * and basically copy the incoming events to the
     * specified {@link ContentHandler}.
     *
     * <p>
     * In doing so, a {@link ValidatorHandler} may modify
     * the events, for example by adding defaulted attributes.
     *
     * <p>
     * A {@link ValidatorHandler} may buffer events to certain
     * extent, but to allow {@link ValidatorHandler} to be used
     * by a parser, the following requirement has to be met.
     *
     * <ol>
     *  <li>When
     *      {@link ContentHandler#startElement(String, String, String, Attributes)},
     *      {@link ContentHandler#endElement(String, String, String)},
     *      {@link ContentHandler#startDocument()}, or
     *      {@link ContentHandler#endDocument()}
     *      are invoked on a {@link ValidatorHandler},
     *      the same method on the user-specified {@link ContentHandler}
     *      must be invoked for the same event before the callback
     *      returns.
     *  <li>{@link ValidatorHandler} may not introduce new elements that
     *      were not present in the input.
     *
     *  <li>{@link ValidatorHandler} may not remove attributes that were
     *      present in the input.
     * </ol>
     *
     * <p>
     * When a callback method on the specified {@link ContentHandler}
     * throws an exception, the same exception object must be thrown
     * from the {@link ValidatorHandler}. The {@link ErrorHandler}
     * should not be notified of such an exception.
     *
     * <p>
     * This method can be called even during a middle of a validation.
     *
```

```
     * @param receiver
     *      A {@link ContentHandler} or a null value.
     */
    public abstract void setContentHandler(ContentHandler receiver);

    /**
     * Gets the {@link ContentHandler} which receives the
     * augmented validation result.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #getContentHandler()} method, or null
     *      if that method has never been called since this {@link ValidatorHandler}
     *      has created.
     *
     * @see #setContentHandler(ContentHandler)
     */
    public abstract ContentHandler getContentHandler();

    /**
     * Sets the {@link ErrorHandler} to receive errors encountered
     * during the validation.
     *
     * <p>
     * Error handler can be used to customize the error handling process
     * during a validation. When an {@link ErrorHandler} is set,
     * errors found during the validation will be first sent
     * to the {@link ErrorHandler}.
     *
     * <p>
     * The error handler can abort further validation immediately
     * by throwing {@link org.xml.sax.SAXException} from the handler. Or for example
     * it can print an error to the screen and try to continue the
     * validation by returning normally from the {@link ErrorHandler}
     *
     * <p>
     * If any {@link Throwable} is thrown from an {@link ErrorHandler},
     * the same {@link Throwable} object will be thrown toward the
     * root of the call stack.
     *
     * <p>
     * {@link ValidatorHandler} is not allowed to
     * throw {@link org.xml.sax.SAXException} without first reporting it to
     * {@link ErrorHandler}.
     *
     * <p>
     * When the {@link ErrorHandler} is null, the implementation will
     * behave as if the following {@link ErrorHandler} is set:
     * <pre>
     * class DraconianErrorHandler implements {@link ErrorHandler} {
     *     public void fatalError( {@link org.xml.sax.SAXParseException} e ) throws {@link org.xml.
         sax.SAXException} {
     *         throw e;
```

```
     *     }
     *     public void error( {@link org.xml.sax.SAXParseException} e ) throws {@link org.xml.sax.
         SAXException} {
     *         throw e;
     *     }
     *     public void warning( {@link org.xml.sax.SAXParseException} e ) throws {@link org.xml.sax
         .SAXException} {
     *         // noop
     *     }
     * }
     * </pre>
     *
     * <p>
     * When a new {@link ValidatorHandler} object is created, initially
     * this field is set to null.
     *
     * @param   errorHandler
     *      A new error handler to be set. This parameter can be null.
     */
    public abstract void setErrorHandler(ErrorHandler errorHandler);

    /**
     * Gets the current {@link ErrorHandler} set to this {@link ValidatorHandler}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setErrorHandler(ErrorHandler)} method, or null
     *      if that method has never been called since this {@link ValidatorHandler}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract ErrorHandler getErrorHandler();

    /**
     * Sets the {@link LSResourceResolver} to customize
     * resource resolution while in a validation episode.
     *
     * <p>
     * {@link ValidatorHandler} uses a {@link LSResourceResolver}
     * when it needs to locate external resources while a validation,
     * although exactly what constitutes "locating external resources" is
     * up to each schema language.
     *
     * <p>
     * When the {@link LSResourceResolver} is null, the implementation will
     * behave as if the following {@link LSResourceResolver} is set:
     * <pre>
     * class DumbLSResourceResolver implements {@link LSResourceResolver} {
     *     public {@link org.w3c.dom.ls.LSInput} resolveResource(
     *         String publicId, String systemId, String baseURI) {
     *
     *         return null; // always return null
```

```
     *     }
     * }
     * </pre>
     *
     * <p>
     * If a {@link LSResourceResolver} throws a {@link RuntimeException}
     *  (or instances of its derived classes),
     * then the {@link ValidatorHandler} will abort the parsing and
     * the caller of the <code>validate</code> method will receive
     * the same {@link RuntimeException}.
     *
     * <p>
     * When a new {@link ValidatorHandler} object is created, initially
     * this field is set to null.
     *
     * @param   resourceResolver
     *      A new resource resolver to be set. This parameter can be null.
     */
    public abstract void setResourceResolver(LSResourceResolver resourceResolver);

    /**
     * Gets the current {@link LSResourceResolver} set to this {@link ValidatorHandler}.
     *
     * @return
     *      This method returns the object that was last set through
     *      the {@link #setResourceResolver(LSResourceResolver)} method, or null
     *      if that method has never been called since this {@link ValidatorHandler}
     *      has created.
     *
     * @see #setErrorHandler(ErrorHandler)
     */
    public abstract LSResourceResolver getResourceResolver();

    /**
     * Obtains the {@link TypeInfoProvider} implementation of this
     * {@link ValidatorHandler}.
     *
     * <p>
     * The obtained {@link TypeInfoProvider} can be queried during a parse
     * to access the type information determined by the validator.
     *
     * <p>
     * Some schema languages do not define the notion of type,
     * for those languages, this method may not be supported.
     * However, to be compliant with this specification, implementations
     * for W3C XML Schema 1.0 must support this operation.
     *
     * @return
     *      null if the validator / schema language does not support
     *      the notion of {@link org.w3c.dom.TypeInfo}.
     *      Otherwise a non-null valid {@link TypeInfoProvider}.
     */
    public abstract TypeInfoProvider getTypeInfoProvider();
```

```


    /**
     * Look up the value of a feature flag.
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for a {@link ValidatorHandler} to recognize a feature name but
     * temporarily be unable to return its value.
     * Some feature values may be available only in specific
     * contexts, such as before, during, or after a validation.
     *
     * <p>Implementors are free (and encouraged) to invent their own features,
     * using names built on their own URIs.</p>
     *
     * @param name The feature name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link ValidatorHandler} recognizes the feature name but
     *   cannot determine its value at this time.
     * @throws NullPointerException When <code>name</code> is <code>null</code>.
     *
     * @see #setFeature(String, boolean)
     */
    public boolean getFeature(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException();
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * <p>Set a feature for this <code>ValidatorHandler</code>.</p>
     *
     * <p>Feature can be used to control the way a
     * {@link ValidatorHandler} parses schemas. The feature name is
     * any fully-qualified URI. It is possible for a
     * {@link SchemaFactory} to
     * expose a feature value but to be unable to change the current
     * value. Some feature values may be immutable or mutable only in
     * specific contexts, such as before, during, or after a
     * validation.</p>
     *
     * <p>All implementations are required to support the {@link javax.xml.XMLConstants#
         FEATURE_SECURE_PROCESSING} feature.
     * When the feature is:</p>
     * <ul>
```

```
     *   <li>
     *     <code>true</code>: the implementation will limit XML processing to conform to implementation limits.
     *     Examples include enity expansion limits and XML Schema constructs that would consume large amounts of resources.
     *     If XML processing is limited for security reasons, it will be reported via a call to the registered
     *    {@link ErrorHandler#fatalError(SAXParseException exception)}.
     *     See {@link #setErrorHandler(ErrorHandler errorHandler)}.
     *   </li>
     *   <li>
     *     <code>false</code>: the implementation will processing XML according to the XML specifications without
     *     regard to possible implementation limits.
     *   </li>
     * </ul>
     *
     * @param name The feature name, which is a non-null fully-qualified URI.
     * @param value The requested value of the feature (true or false).
     *
     * @throws SAXNotRecognizedException If the feature
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link ValidatorHandler} recognizes the feature name but
     *   cannot set the requested value.
     * @throws NullPointerException When <code>name</code> is <code>null</code>.
     *
     * @see #getFeature(String)
     */
    public void setFeature(String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException();
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * Set the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link ValidatorHandler} to recognize a property name but
     * to be unable to change the current value.
     * Some property values may be immutable or mutable only
     * in specific contexts, such as before, during, or after
     * a validation.</p>
     *
     * <p>{@link ValidatorHandler}s are not required to recognize setting
     * any specific property names.</p>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
```

```
     * @param object The requested value for the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   {@link ValidatorHandler} recognizes the property name but
     *   cannot set the requested value.
     * @throws NullPointerException When <code>name</code> is <code>null</code>.
     */
    public void setProperty(String name, Object object)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException();
        }

        throw new SAXNotRecognizedException(name);
    }

    /**
     * Look up the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for a {@link ValidatorHandler} to recognize a property name but
     * temporarily be unable to return its value.
     * Some property values may be available only in specific
     * contexts, such as before, during, or after a validation.</p>
     *
     * <p>{@link ValidatorHandler}s are not required to recognize any specific
     * property names.</p>
     *
     * <p>Implementors are free (and encouraged) to invent their own properties,
     * using names built on their own URIs.</p>
     *
     * @param name The property name, which is a non-null fully-qualified URI.
     *
     * @return The current value of the property.
     *
     * @throws SAXNotRecognizedException If the property
     *   value can't be assigned or retrieved.
     * @throws SAXNotSupportedException When the
     *   XMLReader recognizes the property name but
     *   cannot determine its value at this time.
     * @throws NullPointerException When <code>name</code> is <code>null</code>.
     *
     * @see #setProperty(String, Object)
     */
    public Object getProperty(String name)
        throws SAXNotRecognizedException, SAXNotSupportedException {

        if (name == null) {
            throw new NullPointerException();
        }
```

```

        throw new SAXNotRecognizedException(name);
    }
}
```

27 Xml Ws (javax.xml.ws package)

27.1 Action.java

Secuencia 260: javax/xml/ws/Action.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * The <code>Action</code> annotation allows explicit association of a
 * WS-Addressing <code>Action</code> message addressing property with
 * <code>input</code>, <code>output</code>, and
 * <code>fault</code> messages of the mapped WSDL operation.
 * <p>
 * This annotation can be specified on each method of a service endpoint interface.
 * For such a method, the mapped operation in the generated WSDL's
 * <code>wsam:Action</code> attribute on the WSDL <code>input</code>,
 * <code>output</code> and <code>fault</code> messages of the WSDL <code>operation</code>
```

```
* is based upon which attributes of the <code>Action</code> annotation have been specified.
* For the exact computation of <code>wsam:Action</code> values for the messages, refer
* to the algorithm in the JAX-WS specification.
* <p>
* <b>Example 1</b>: Specify explicit values for <code>Action</code> message addressing property
* for <code>input</code> and <code>output</code> messages.
*
* <pre>
* &#64;WebService(targetNamespace="http://example.com/numbers")
* public class AddNumbersImpl {
*     <b>&#64;Action(
*         input="http://example.com/inputAction",
*         output="http://example.com/outputAction")</b>
*     public int addNumbers(int number1, int number2) {
*         return number1 + number2;
*     }
* }
* </pre>
*
* The generated WSDL looks like:
* <pre>
*   &lt;definitions targetNamespace="http://example.com/numbers" ...>
*     ...
*     &lt;portType name="AddNumbersPortType">
*       &lt;operation name="AddNumbers">
*         &lt;input message="tns:AddNumbersInput" name="foo"
*           <b>wsam:Action="http://example.com/inputAction"</b>/>
*         &lt;output message="tns:AddNumbersOutput" name="bar"
*           <b>wsam:Action="http://example.com/outputAction"</b>/>
*       &lt;/operation>
*     &lt;/portType>
*     ...
*   &lt;/definitions>
* </pre>
*
* <p>
* <b>Example 2</b>: Specify explicit value for <code>Action</code> message addressing property
* for only the <code>input</code> message. The <code>wsam:Action</code> values for the
* WSDL <code>output</code> message are computed using the algorithm in the JAX-WS specification.
*
* <pre>
* &#64;WebService(targetNamespace="http://example.com/numbers")
* public class AddNumbersImpl {
*     <b>&#64;Action(input="http://example.com/inputAction")</b>
*     public int addNumbers(int number1, int number2) {
*         return number1 + number2;
*     }
* }
* </pre>
*
* The generated WSDL looks like:
* <pre>
*   &lt;definitions targetNamespace="http://example.com/numbers" ...>
```

```
 *     ...
 *     &lt;portType name="AddNumbersPortType">
 *       &lt;operation name="AddNumbers">
 *         &lt;input message="tns:AddNumbersInput" name="foo"
 *           <b>wsam:Action="http://example.com/inputAction"</b> />
 *         &lt;output message="tns:AddNumbersOutput" name="bar"
 *           <b>wsam:Action="http://example.com/numbers/AddNumbersPortType/AddNumbersResponse"</b>/>
 *       &lt;/operation>
 *     &lt;/portType>
 *     ...
 *   &lt;/definitions>
 * </pre>
 *
 * It is legitimate to specify an explicit value for <code>Action</code> message addressing property for
 * <code>output</code> message only. In this case, <code>wsam:Action</code> value for the
 * WSDL <code>input</code> message is computed using the algorithm in the JAX-WS specification.
 *
 * <p>
 * <b>Example 3</b>: See {@link FaultAction} annotation for an example of
 * how to specify an explicit value for <code>Action</code> message addressing property for the
 * <code>fault</code> message.
 *
 * @see FaultAction
 *
 * @since JAX-WS 2.1
 */

@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Action {
    /**
     * Explicit value of the WS-Addressing <code>Action</code> message addressing property for the <code>input</code>
     * message of the operation.
     */
    String input() default "";

    /**
     * Explicit value of the WS-Addressing <code>Action</code> message addressing property for the <code>output</code>
     * message of the operation.
     */
    String output() default "";

    /**
     * Explicit value of the WS-Addressing <code>Action</code> message addressing property for the <code>fault</code>
     * message(s) of the operation. Each exception that is mapped to a fault and requires an explicit WS-Addressing
     * <code>Action</code> message addressing property, needs to be specified as a value in this
```

```
        property
     * using {@link FaultAction} annotation.
     */
    FaultAction[] fault() default { };
}
```

27.2 AsyncHandler.java

Secuencia 261: javax/xml/ws/AsyncHandler.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

/** The <code>AsyncHandler</code> interface is implemented by
 * clients that wish to receive callback notification of the completion of
 * service endpoint operations invoked asynchronously.
 *
 *  @since JAX-WS 2.0
**/
public interface AsyncHandler<T> {

    /** Called when the response to an asynchronous operation is available.
     *
     * @param res The response to the operation invocation.
     *
    **/
    void handleResponse(Response<T> res);
}
```

27.3 Binding.java

Secuencia 262: javax/xml/ws/Binding.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;


/** The <code>Binding</code> interface is the base interface
 *  for JAX-WS protocol bindings.
 *
 *  @since JAX-WS 2.0
**/
public interface Binding {

   /**
    * Gets a copy of the handler chain for a protocol binding instance.
    * If the returned chain is modified a call to <code>setHandlerChain</code>
    * is required to configure the binding instance with the new chain.
    *
    *  @return java.util.List&lt;Handler> Handler chain
    */
    public java.util.List<javax.xml.ws.handler.Handler> getHandlerChain();

   /**
    * Sets the handler chain for the protocol binding instance.
    *
    *  @param chain    A List of handler configuration entries
    *  @throws WebServiceException On an error in the configuration of
    *                  the handler chain
    *  @throws java.lang.UnsupportedOperationException If this
    *          operation is not supported. This may be done to
```

```
    *          avoid any overriding of a pre-configured handler
    *          chain.
    */
    public void setHandlerChain(java.util.List<javax.xml.ws.handler.Handler> chain);

    /**
     * Get the URI for this binding instance.
     *
     * @return String The binding identifier for the port.
     *    Never returns <code>null</code>
     *
     * @since JAX-WS 2.1
     */
    String getBindingID();
}
```

27.4 BindingProvider.java

Secuencia 263: javax/xml/ws/BindingProvider.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.util.Map;
import javax.xml.ws.wsaddressing.W3CEndpointReference;

/**
 * The <code>BindingProvider</code> interface provides access to the
 * protocol binding and associated context objects for request and
 * response message processing.
```

```
 *
 * @since JAX-WS 2.0
 *
 * @see javax.xml.ws.Binding
 **/
public interface BindingProvider {
    /**
     * Standard property: User name for authentication.
     * <p>Type: <code>java.lang.String</code>
     **/
    public static final String USERNAME_PROPERTY =
            "javax.xml.ws.security.auth.username";

    /**
     * Standard property: Password for authentication.
     * <p>Type: <code>java.lang.String</code>
     **/
    public static final String PASSWORD_PROPERTY =
            "javax.xml.ws.security.auth.password";

    /**
     * Standard property: Target service endpoint address. The
     * URI scheme for the endpoint address specification MUST
     * correspond to the protocol/transport binding for the
     * binding in use.
     * <p>Type: <code>java.lang.String</code>
     **/
    public static final String ENDPOINT_ADDRESS_PROPERTY =
            "javax.xml.ws.service.endpoint.address";

    /**
     * Standard property: This boolean property is used by a service
     * client to indicate whether or not it wants to participate in
     * a session with a service endpoint. If this property is set to
     * <code>true</code>, the service client indicates that it wants the session
     * to be maintained. If set to <code>false</code>, the session is not maintained.
     * The default value for this property is <code>false</code>.
     * <p>Type: <code>java.lang.Boolean</code>
     **/
    public static final String SESSION_MAINTAIN_PROPERTY =
            "javax.xml.ws.session.maintain";

    /**
     * Standard property for SOAPAction. This boolean property
     * indicates whether or not the value of the
     * <code>javax.xml.ws.soap.http.soapaction.uri</code> property
     * is used for the value of the SOAPAction. The
     * default value of this property is <code>false</code> indicating
     * that the
     * <code>javax.xml.ws.soap.http.soapaction.uri</code> property
     * is not used for the value of the SOAPAction, however,
     * if WS-Addressing is enabled, the default value is
     * <code>true</code>.
```

```
     *
     * <p>Type: <code>java.lang.Boolean</code>
     **/
    public static final String SOAPACTION_USE_PROPERTY =
            "javax.xml.ws.soap.http.soapaction.use";

    /**
     * Standard property for SOAPAction. Indicates the SOAPAction
     * URI if the <code>javax.xml.ws.soap.http.soapaction.use</code>
     * property is set to <code>true</code>. If WS-Addressing
     * is enabled, this value will also be used for the value of the
     * WS-Addressing Action header.  If this property is not set,
     * the default SOAPAction and WS-Addressing Action will be sent.
     *
     * <p>Type: <code>java.lang.String</code>
     **/
    public static final String SOAPACTION_URI_PROPERTY =
            "javax.xml.ws.soap.http.soapaction.uri";

    /**
     * Get the context that is used to initialize the message context
     * for request messages.
     *
     * Modifications to the request context do not affect the message context of
     * either synchronous or asynchronous operations that have already been
     * started.
     *
     * @return The context that is used in processing request messages.
     **/
    Map<String, Object> getRequestContext();

    /**
     * Get the context that resulted from processing a response message.
     *
     * The returned context is for the most recently completed synchronous
     * operation. Subsequent synchronous operation invocations overwrite the
     * response context. Asynchronous operations return their response context
     * via the Response interface.
     *
     * @return The context that resulted from processing the latest
     * response messages.
     **/
    Map<String, Object> getResponseContext();

    /**
     * Get the Binding for this binding provider.
     *
     * @return The Binding for this binding provider.
     **/
    Binding getBinding();
```

```
    /**
     * Returns the <code>EndpointReference</code> associated with
     * this <code>BindingProvider</code> instance.
     * <p>
     * If the Binding for this <code>bindingProvider</code> is
     * either SOAP1.1/HTTP or SOAP1.2/HTTP, then a
     * <code>W3CEndpointReference</code> MUST be returned.
     *
     * @return EndpointReference of the target endpoint associated with this
     * <code>BindingProvider</code> instance.
     *
     * @throws java.lang.UnsupportedOperationException If this
     * <code>BindingProvider</code> uses the XML/HTTP binding.
     *
     * @see W3CEndpointReference
     *
     * @since JAX-WS 2.1
     */
    public EndpointReference getEndpointReference();


    /**
     * Returns the <code>EndpointReference</code> associated with
     * this <code>BindingProvider</code> instance.  The instance
     * returned will be of type <code>clazz</code>.
     *
     * @param clazz Specifies the type of <code>EndpointReference</code>
     * that MUST be returned.

     * @return EndpointReference of the target endpoint associated with this
     * <code>BindingProvider</code> instance. MUST be of type
     * <code>clazz</code>.

     * @throws WebServiceException If the Class <code>clazz</code>
     * is not supported by this implementation.
     * @throws java.lang.UnsupportedOperationException If this
     * <code>BindingProvider</code> uses the XML/HTTP binding.
     *
     * @since JAX-WS 2.1
     */
    public <T extends EndpointReference> T getEndpointReference(Class<T> clazz);
}
```

27.5 BindingType.java

Secuencia 264: javax/xml/ws/BindingType.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

/**
 *  The <code>BindingType</code> annotation is used to
 *  specify the binding to use for a web service
 *  endpoint implementation class.
 *  <p>
 *  This annotation may be overriden programmatically or via
 *  deployment descriptors, depending on the platform in use.
 *
 *  @since JAX-WS 2.0
 *
 **/
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface BindingType {
     /**
      * A binding identifier (a URI).
      * If not specified, the default is the SOAP 1.1 / HTTP binding.
      * <p>
      * See the <code>SOAPBinding</code> and <code>HTTPBinding</code>
      * for the definition of the standard binding identifiers.
      *
      * @see javax.xml.ws.Binding
      * @see javax.xml.ws.soap.SOAPBinding#SOAP11HTTP_BINDING
      * @see javax.xml.ws.soap.SOAPBinding#SOAP12HTTP_BINDING
      * @see javax.xml.ws.http.HTTPBinding#HTTP_BINDING
      */
```

```
    String value() default "" ;
}
```

27.6 Dispatch.java

Secuencia 265: javax/xml/ws/Dispatch.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.util.concurrent.Future;

/** The <code>Dispatch</code> interface provides support
 *  for the dynamic invocation of a service endpoint operations. The
 *  <code>javax.xml.ws.Service</code>
 *  class acts as a factory for the creation of <code>Dispatch</code>
 *  instances.
 *
 *  @since JAX-WS 2.0
**/
public interface Dispatch<T> extends BindingProvider {

    /** Invoke a service operation synchronously.
     *
     * The client is responsible for ensuring that the <code>msg</code> object
     * when marshalled is formed according to the requirements of the protocol
     * binding in use.
     *
     * @param msg An object that will form the message or payload of
     *     the message used to invoke the operation.
```

```
     * @return The response message or message payload to the
     *     operation invocation.
     * @throws WebServiceException If a fault occurs during communication with
     *     the service
     * @throws WebServiceException If there is any error in the configuration of
     *     the <code>Dispatch</code> instance
    **/
    public T invoke(T msg);

    /** Invoke a service operation asynchronously.  The
     *  method returns without waiting for the response to the operation
     *  invocation, the results of the operation are obtained by polling the
     *  returned <code>Response</code>.
     * <p>
     * The client is responsible for ensuring that the <code>msg</code> object
     * when marshalled is formed according to the requirements of the protocol
     * binding in use.
     *
     * @param msg An object that will form the message or payload of
     *     the message used to invoke the operation.
     * @return The response message or message payload to the
     *     operation invocation.
     * @throws WebServiceException If there is any error in the configuration of
     *     the <code>Dispatch</code> instance
    **/
    public Response<T> invokeAsync(T msg);

    /** Invoke a service operation asynchronously. The
     *  method returns without waiting for the response to the operation
     *  invocation, the results of the operation are communicated to the client
     *  via the passed in <code>handler</code>.
     * <p>
     * The client is responsible for ensuring that the <code>msg</code> object
     * when marshalled is formed according to the requirements of the protocol
     * binding in use.
     *
     * @param msg An object that will form the message or payload of
     *     the message used to invoke the operation.
     * @param handler The handler object that will receive the
     *     response to the operation invocation.
     * @return A <code>Future</code> object that may be used to check the status
     *     of the operation invocation. This object MUST NOT be used to try to
     *     obtain the results of the operation - the object returned from
     *     <code>Future&lt;?>.get()</code> is implementation dependent
     *     and any use of it will result in non-portable behaviour.
     * @throws WebServiceException If there is any error in the configuration of
     *     the <code>Dispatch</code> instance
    **/
    public Future<?> invokeAsync(T msg, AsyncHandler<T> handler);

    /** Invokes a service operation using the one-way
     *  interaction mode. The operation invocation is logically non-blocking,
     *  subject to the capabilities of the underlying protocol, no results
```

```
     *  are returned. When
     *  the protocol in use is SOAP/HTTP, this method MUST block until
     *  an HTTP response code has been received or an error occurs.
     * <p>
     * The client is responsible for ensuring that the <code>msg</code> object
     * when marshalled is formed according to the requirements of the protocol
     * binding in use.
     *
     * @param msg An object that will form the message or payload of
     *     the message used to invoke the operation.
     * @throws WebServiceException If there is any error in the configuration of
     *     the <code>Dispatch</code> instance or if an error occurs during the
     *     invocation.
    **/
    public void invokeOneWay(T msg);
}
```

27.7 Endpoint.java

Secuencia 266: javax/xml/ws/Endpoint.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.util.List;
import java.util.Map;
import javax.xml.ws.spi.Provider;
import javax.xml.ws.spi.http.HttpContext;
import javax.xml.ws.wsaddressing.W3CEndpointReference;
import org.w3c.dom.Element;
```

```


/**
 * A Web service endpoint.
 *
 * <p>Endpoints are created using the static methods defined in this
 * class. An endpoint is always tied to one <code>Binding</code>
 * and one implementor, both set at endpoint creation time.
 *
 * <p>An endpoint is either in a published or an unpublished state.
 * The <code>publish</code> methods can be used to start publishing
 * an endpoint, at which point it starts accepting incoming requests.
 * Conversely, the <code>stop</code> method can be used to stop
 * accepting incoming requests and take the endpoint down.
 * Once stopped, an endpoint cannot be published again.
 *
 * <p>An <code>Executor</code> may be set on the endpoint in order
 * to gain better control over the threads used to dispatch incoming
 * requests. For instance, thread pooling with certain parameters
 * can be enabled by creating a <code>ThreadPoolExecutor</code> and
 * registering it with the endpoint.
 *
 * <p>Handler chains can be set using the contained <code>Binding</code>.
 *
 * <p>An endpoint may have a list of metadata documents, such as WSDL
 * and XMLSchema documents, bound to it. At publishing time, the
 * JAX-WS implementation will try to reuse as much of that metadata
 * as possible instead of generating new ones based on the annotations
 * present on the implementor.
 *
 * @since JAX-WS 2.0
 *
 * @see javax.xml.ws.Binding
 * @see javax.xml.ws.BindingType
 * @see javax.xml.ws.soap.SOAPBinding
 * @see java.util.concurrent.Executor
 *
 **/
public abstract class Endpoint {

    /** Standard property: name of WSDL service.
     *  <p>Type: javax.xml.namespace.QName
     **/
    public static final String WSDL_SERVICE = "javax.xml.ws.wsdl.service";

    /** Standard property: name of WSDL port.
     *  <p>Type: javax.xml.namespace.QName
     **/
    public static final String WSDL_PORT = "javax.xml.ws.wsdl.port";


    /**
     * Creates an endpoint with the specified implementor object. If there is
```

```
     * a binding specified via a BindingType annotation then it MUST be used else
     * a default of SOAP 1.1 / HTTP binding MUST be used.
     * <p>
     * The newly created endpoint may be published by calling
     * one of the {@link javax.xml.ws.Endpoint#publish(String)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods.
     *
     *
     * @param implementor The endpoint implementor.
     *
     * @return The newly created endpoint.
     *
     **/
    public static Endpoint create(Object implementor) {
        return create(null, implementor);
    }

    /**
     * Creates an endpoint with the specified implementor object and web
     * service features. If there is a binding specified via a BindingType
     * annotation then it MUST be used else a default of SOAP 1.1 / HTTP
     * binding MUST be used.
     * <p>
     * The newly created endpoint may be published by calling
     * one of the {@link javax.xml.ws.Endpoint#publish(String)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods.
     *
     *
     * @param implementor The endpoint implementor.
     * @param features A list of WebServiceFeature to configure on the
     *        endpoint. Supported features not in the <code>features
     *        </code> parameter will have their default values.
     *
     *
     * @return The newly created endpoint.
     * @since JAX-WS 2.2
     *
     */
    public static Endpoint create(Object implementor, WebServiceFeature ... features) {
        return create(null, implementor, features);
    }

    /**
     * Creates an endpoint with the specified binding type and
     * implementor object.
     * <p>
     * The newly created endpoint may be published by calling
     * one of the {@link javax.xml.ws.Endpoint#publish(String)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods.
     *
     * @param bindingId A URI specifying the binding to use. If the bindingID is
     * <code>null</code> and no binding is specified via a BindingType
     * annotation then a default SOAP 1.1 / HTTP binding MUST be used.
```

```
     *
     * @param implementor The endpoint implementor.
     *
     * @return The newly created endpoint.
     *
     **/
    public static Endpoint create(String bindingId, Object implementor) {
        return Provider.provider().createEndpoint(bindingId, implementor);
    }

    /**
     * Creates an endpoint with the specified binding type,
     * implementor object, and web service features.
     * <p>
     * The newly created endpoint may be published by calling
     * one of the {@link javax.xml.ws.Endpoint#publish(String)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods.
     *
     * @param bindingId A URI specifying the binding to use. If the bindingID is
     * <code>null</code> and no binding is specified via a BindingType
     * annotation then a default SOAP 1.1 / HTTP binding MUST be used.
     *
     * @param implementor The endpoint implementor.
     *
     * @param features A list of WebServiceFeature to configure on the
     *        endpoint. Supported features not in the <code>features
     *        </code> parameter will have their default values.
     *
     * @return The newly created endpoint.
     * @since JAX-WS 2.2
     */
    public static Endpoint create(String bindingId, Object implementor, WebServiceFeature ...
        features) {
        return Provider.provider().createEndpoint(bindingId, implementor, features);
    }

    /**
     * Returns the binding for this endpoint.
     *
     * @return The binding for this endpoint
     **/
    public abstract Binding getBinding();

    /**
     * Returns the implementation object for this endpoint.
     *
     * @return The implementor for this endpoint
     **/
    public abstract Object getImplementor();

    /**
     * Publishes this endpoint at the given address.
     * The necessary server infrastructure will be created and
```

```
     * configured by the JAX-WS implementation using some default configuration.
     * In order to get more control over the server configuration, please
     * use the {@link javax.xml.ws.Endpoint#publish(Object)} method instead.
     *
     * @param address A URI specifying the address to use. The address
     *        MUST be compatible with the binding specified at the
     *        time the endpoint was created.
     *
     * @throws java.lang.IllegalArgumentException
     *          If the provided address URI is not usable
     *          in conjunction with the endpoint's binding.
     *
     * @throws java.lang.IllegalStateException
     *          If the endpoint has been published already or it has been stopped.
     *
     * @throws java.lang.SecurityException
     *          If a <code>java.lang.SecurityManger</code>
     *          is being used and the application doesn't have the
     *          <code>WebServicePermission("publishEndpoint")</code> permission.
     **/
    public abstract void publish(String address);

    /**
     * Creates and publishes an endpoint for the specified implementor
     * object at the given address.
     * <p>
     * The necessary server infrastructure will be created and
     * configured by the JAX-WS implementation using some default configuration.
     *
     * In order to get more control over the server configuration, please
     * use the {@link javax.xml.ws.Endpoint#create(String,Object)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods instead.
     *
     * @param address A URI specifying the address and transport/protocol
     *        to use. A http: URI MUST result in the SOAP 1.1/HTTP
     *        binding being used. Implementations may support other
     *        URI schemes.
     * @param implementor The endpoint implementor.
     *
     * @return The newly created endpoint.
     *
     * @throws java.lang.SecurityException
     *          If a <code>java.lang.SecurityManger</code>
     *          is being used and the application doesn't have the
     *          <code>WebServicePermission("publishEndpoint")</code> permission.
     *
     **/
    public static Endpoint publish(String address, Object implementor) {
        return Provider.provider().createAndPublishEndpoint(address, implementor);
    }

    /**
     * Creates and publishes an endpoint for the specified implementor
```

```
     * object at the given address. The created endpoint is configured
     * with the web service features.
     * <p>
     * The necessary server infrastructure will be created and
     * configured by the JAX-WS implementation using some default configuration.
     *
     * In order to get more control over the server configuration, please
     * use the {@link javax.xml.ws.Endpoint#create(String,Object)} and
     * {@link javax.xml.ws.Endpoint#publish(Object)} methods instead.
     *
     * @param address A URI specifying the address and transport/protocol
     *        to use. A http: URI MUST result in the SOAP 1.1/HTTP
     *        binding being used. Implementations may support other
     *        URI schemes.
     * @param implementor The endpoint implementor.
     * @param features A list of WebServiceFeature to configure on the
     *        endpoint. Supported features not in the <code>features
     *        </code> parameter will have their default values.
     * @return The newly created endpoint.
     *
     * @throws java.lang.SecurityException
     *          If a <code>java.lang.SecurityManger</code>
     *          is being used and the application doesn't have the
     *          <code>WebServicePermission("publishEndpoint")</code> permission.
     * @since JAX-WS 2.2
     */
    public static Endpoint publish(String address, Object implementor, WebServiceFeature ...
        features) {
        return Provider.provider().createAndPublishEndpoint(address, implementor, features);
    }


    /**
     * Publishes this endpoint at the provided server context.
     * A server context encapsulates the server infrastructure
     * and addressing information for a particular transport.
     * For a call to this method to succeed, the server context
     * passed as an argument to it MUST be compatible with the
     * endpoint's binding.
     *
     * @param serverContext An object representing a server
     *           context to be used for publishing the endpoint.
     *
     * @throws java.lang.IllegalArgumentException
     *              If the provided server context is not
     *              supported by the implementation or turns
     *              out to be unusable in conjunction with the
     *              endpoint's binding.
     *
     * @throws java.lang.IllegalStateException
     *         If the endpoint has been published already or it has been stopped.
     *
     * @throws java.lang.SecurityException
```

```
     *          If a <code>java.lang.SecurityManger</code>
     *          is being used and the application doesn't have the
     *          <code>WebServicePermission("publishEndpoint")</code> permission.
     **/
    public abstract void publish(Object serverContext);

    /**
     * Publishes this endpoint at the provided server context.
     * A server context encapsulates the server infrastructure
     * and addressing information for a particular transport.
     * For a call to this method to succeed, the server context
     * passed as an argument to it MUST be compatible with the
     * endpoint's binding.
     *
     * <p>
     * This is meant for container developers to publish the
     * the endpoints portably and not intended for the end
     * developers.
     *
     *
     * @param serverContext An object representing a server
     *           context to be used for publishing the endpoint.
     *
     * @throws java.lang.IllegalArgumentException
     *              If the provided server context is not
     *              supported by the implementation or turns
     *              out to be unusable in conjunction with the
     *              endpoint's binding.
     *
     * @throws java.lang.IllegalStateException
     *         If the endpoint has been published already or it has been stopped.
     *
     * @throws java.lang.SecurityException
     *          If a <code>java.lang.SecurityManger</code>
     *          is being used and the application doesn't have the
     *          <code>WebServicePermission("publishEndpoint")</code> permission.
     * @since JAX-WS 2.2
     */
    public void publish(HttpContext serverContext) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }

    /**
     * Stops publishing this endpoint.
     *
     * If the endpoint is not in a published state, this method
     * has no effect.
     *
     **/
    public abstract void stop();

    /**
```

```
     * Returns true if the endpoint is in the published state.
     *
     * @return <code>true</code> if the endpoint is in the published state.
     **/
    public abstract boolean isPublished();

    /**
     * Returns a list of metadata documents for the service.
     *
     * @return <code>List&lt;javax.xml.transform.Source&gt;</code> A list of metadata documents for
          the service
     **/
    public abstract List<javax.xml.transform.Source> getMetadata();

    /**
     * Sets the metadata for this endpoint.
     *
     * @param metadata A list of XML document sources containing
     *           metadata information for the endpoint (e.g.
     *           WSDL or XML Schema documents)
     *
     * @throws java.lang.IllegalStateException  If the endpoint
     *         has already been published.
     **/
    public abstract void setMetadata(List<javax.xml.transform.Source> metadata);

    /**
     * Returns the executor for this <code>Endpoint</code>instance.
     *
     * The executor is used to dispatch an incoming request to
     * the implementor object.
     *
     * @return The <code>java.util.concurrent.Executor</code> to be
     *         used to dispatch a request.
     *
     * @see java.util.concurrent.Executor
     **/
    public abstract java.util.concurrent.Executor getExecutor();

    /**
     * Sets the executor for this <code>Endpoint</code> instance.
     *
     * The executor is used to dispatch an incoming request to
     * the implementor object.
     *
     * If this <code>Endpoint</code> is published using the
     * <code>publish(Object)</code> method and the specified server
     * context defines its own threading behavior, the executor
     * may be ignored.
     *
     * @param executor The <code>java.util.concurrent.Executor</code>
     *        to be used to dispatch a request.
     *
```

```
     * @throws SecurityException  If the instance does not support
     *         setting an executor for security reasons (e.g. the
     *         necessary permissions are missing).
     *
     * @see java.util.concurrent.Executor
     **/
    public abstract void setExecutor(java.util.concurrent.Executor executor);


    /**
     * Returns the property bag for this <code>Endpoint</code> instance.
     *
     * @return Map&lt;String,Object&gt; The property bag
     *         associated with this instance.
     **/
    public abstract Map<String,Object> getProperties();

    /**
     * Sets the property bag for this <code>Endpoint</code> instance.
     *
     * @param properties The property bag associated with
     *        this instance.
     **/
    public abstract void setProperties(Map<String,Object> properties);

    /**
     * Returns the <code>EndpointReference</code> associated with
     * this <code>Endpoint</code> instance.
     * <p>
     * If the Binding for this <code>bindingProvider</code> is
     * either SOAP1.1/HTTP or SOAP1.2/HTTP, then a
     * <code>W3CEndpointReference</code> MUST be returned.
     *
     * @param referenceParameters Reference parameters to be associated with the
     * returned <code>EndpointReference</code> instance.
     * @return EndpointReference of this <code>Endpoint</code> instance.
     * If the returned <code>EndpointReference</code> is of type
     * <code>W3CEndpointReference</code> then it MUST contain the
     * the specified <code>referenceParameters</code>.

     * @throws WebServiceException If any error in the creation of
     * the <code>EndpointReference</code> or if the <code>Endpoint</code> is
     * not in the published state.
     * @throws UnsupportedOperationException If this <code>BindingProvider</code>
     * uses the XML/HTTP binding.
     *
     * @see W3CEndpointReference
     *
     * @since JAX-WS 2.1
     **/
    public abstract EndpointReference getEndpointReference(Element... referenceParameters);


```

```
    /**
     * Returns the <code>EndpointReference</code> associated with
     * this <code>Endpoint</code> instance.
     *
     * @param clazz Specifies the type of EndpointReference  that MUST be returned.
     * @param referenceParameters Reference parameters to be associated with the
     * returned <code>EndpointReference</code> instance.
     * @return EndpointReference of type <code>clazz</code> of this
     * <code>Endpoint</code> instance.
     * If the returned <code>EndpointReference</code> is of type
     * <code>W3CEndpointReference</code> then it MUST contain the
     * the specified <code>referenceParameters</code>.

     * @throws WebServiceException If any error in the creation of
     * the <code>EndpointReference</code> or if the <code>Endpoint</code> is
     * not in the published state or if the <code>clazz</code> is not a supported
     * <code>EndpointReference</code> type.
     * @throws UnsupportedOperationException If this <code>BindingProvider</code>
     * uses the XML/HTTP binding.
     *
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T extends EndpointReference> T getEndpointReference(Class<T> clazz,
            Element... referenceParameters);

    /**
     * By settng a <code>EndpointContext</code>, JAX-WS runtime knows about
     * addresses of other endpoints in an application. If multiple endpoints
     * share different ports of a WSDL, then the multiple port addresses
     * are patched when the WSDL is accessed.
     *
     * <p>
     * This needs to be set before publishing the endpoints.
     *
     * @param ctxt that is shared for multiple endpoints
     * @throws java.lang.IllegalStateException
     *        If the endpoint has been published already or it has been stopped.
     *
     * @since JAX-WS 2.2
     */
    public void setEndpointContext(EndpointContext ctxt) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }
}
```

27.8 EndpointContext.java

Secuencia 267: javax/xml/ws/EndpointContext.java

```
/*
 * Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.ws.Endpoint;
import java.util.Set;

/**
 * <code>EndpointContext</code> allows multiple endpoints in an application
 * to share any information. For example, servlet application's war may
 * contain multiple endpoints and these endpoints can get addresses of each
 * other by sharing this context. If multiple endpoints share different
 * ports of a WSDL, then the multiple port addresses can be patched
 * when the WSDL is accessed. It also allows all endpoints to share any
 * other runtime information.
 *
 * <p>
 * This needs to be set by using {@link Endpoint#setEndpointContext}
 * before {@link Endpoint#publish} methods.
 *
 * @author Jitendra Kotamraju
 * @since JAX-WS 2.2
 */
public abstract class EndpointContext {

    /**
     * This gives list of endpoints in an application. For e.g in
     * servlet container, a war file may contain multiple endpoints.
     * In case of http, these endpoints are hosted on the same http
     * server.
     *
     * @return list of all endpoints in an application
     */
```

```
    public abstract Set<Endpoint> getEndpoints();

}
```

27.9 EndpointReference.java

Secuencia 268: javax/xml/ws/EndpointReference.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.bind.annotation.XmlTransient;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamResult;
import javax.xml.ws.spi.Provider;
import javax.xml.ws.wsaddressing.W3CEndpointReference;
import java.io.StringWriter;

/**
 * This class represents an WS-Addressing EndpointReference
 * which is a remote reference to a web service endpoint.
 * See <a href="http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/">
 * Web Services Addressing 1.0 - Core</a>
 * for more information on WS-Addressing EndpointReferences.
 * <p>
 * This class is immutable as the typical web service developer
 * need not be concerned with its contents.  The web service
 * developer should use this class strictly as a mechanism to
 * reference a remote web service endpoint. See the {@link Service} APIs
```

```
 * that clients can use to that utilize an <code>EndpointReference</code>.
 * See the {@link javax.xml.ws.Endpoint}, and
 * {@link javax.xml.ws.BindingProvider} APIs on how
 * <code>EndpointReferences</code> can be created for published
 * endpoints.
 * <p>
 * Concrete implementations of this class will represent
 * an <code>EndpointReference</code> for a particular version of Addressing.
 * For example the {@link W3CEndpointReference} is for use
 * with W3C Web Services Addressing 1.0 - Core Recommendation.
 * If JAX-WS implementors need to support different versions
 * of addressing, they should write their own
 * <code>EndpointReference</code> subclass for that version.
 * This will allow a JAX-WS implementation to create
 * a vendor specific <code>EndpointReferences</code> that the
 * vendor can use to flag a different version of
 * addressing.
 * <p>
 * Web service developers that wish to pass or return
 * <code>EndpointReference</code> in Java methods in an
 * SEI should use
 * concrete instances of an <code>EndpointReference</code> such
 * as the <code>W3CEndpointReference</code>.  This way the
 * schema mapped from the SEI will be more descriptive of the
 * type of endpoint reference being passed.
 * <p>
 * JAX-WS implementors are expected to extract the XML infoset
 * from an <CODE>EndpointReferece</CODE> using the
 * <code>{@link EndpointReference#writeTo}</code>
 * method.
 * <p>
 * JAXB will bind this class to xs:anyType. If a better binding
 * is desired, web services developers should use a concrete
 * subclass such as {@link W3CEndpointReference}.
 *
 * @see W3CEndpointReference
 * @see Service
 * @since JAX-WS 2.1
 */
@XmlTransient // to treat this class like Object as far as databinding is concerned (proposed JAXB
    2.1 feature)
public abstract class EndpointReference {
    //
    //Default constructor to be only called by derived types.
    //
    protected EndpointReference(){}

    /**
     * Factory method to read an EndpointReference from the infoset contained in
     * <code>eprInfoset</code>. This method delegates to the vendor specific
     * implementation of the {@link javax.xml.ws.spi.Provider#readEndpointReference} method.
     *
     * @param eprInfoset The <code>EndpointReference</code> infoset to be unmarshalled
```

```
     *
     * @return the EndpointReference unmarshalled from <code>eprInfoset</code>
     *    never <code>null</code>
     * @throws WebServiceException
     *    if an error occurs while creating the
     *    <code>EndpointReference</code> from the <CODE>eprInfoset</CODE>
     * @throws java.lang.IllegalArgumentException
     *     if the <code>null</code> <code>eprInfoset</code> value is given.
     */
    public static EndpointReference readFrom(Source eprInfoset) {
        return Provider.provider().readEndpointReference(eprInfoset);
    }

    /**
     * write this <code>EndpointReference</code> to the specified infoset format
     *
     * @param result for writing infoset
     * @throws WebServiceException
     *   if there is an error writing the
     *   <code>EndpointReference</code> to the specified <code>result</code>.
     *
     * @throws java.lang.IllegalArgumentException
     *      If the <code>null</code> <code>result</code> value is given.
     */
    public abstract void writeTo(Result result);


    /**
     * The <code>getPort</code> method returns a proxy. If there
     * are any reference parameters in the
     * <code>EndpointReference</code> instance, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The parameter  <code>serviceEndpointInterface</code> specifies
     * the service endpoint interface that is supported by the
     * returned proxy.
     * The <code>EndpointReference</code> instance specifies the
     * endpoint that will be invoked by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly from
     * the WSDL Metadata from this <code>EndpointReference</code> or from
     * annotations on the <code>serviceEndpointInterface</code>.  For this method
     * to successfully return a proxy, WSDL metadata MUST be available and the
     * <code>EndpointReference</code> instance MUST contain an implementation understood
     * <code>serviceName</code> metadata.
     * <p>
     * Because this port is not created from a <code>Service</code> object, handlers
     * will not automatically be configured, and the <code>HandlerResolver</code>
     * and <code>Executor</code> cannot be get or set for this port. The
     * <code>BindingProvider().getBinding().setHandlerChain()</code>
     * method can be used to manually configure handlers for this port.
     *
```

```
     *
     * @param serviceEndpointInterface Service endpoint interface
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that supports the
     *                  specified service endpoint interface
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method
     *                  <LI>If this
     *                      <code>endpointReference</code>
     *                      is invalid
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified
     *                  <LI>If a feature is enabled that is not compatible with
     *                      this port or is unsupported.
     *                   </UL>
     *
     * @see java.lang.reflect.Proxy
     * @see WebServiceFeature
     **/
    public <T> T getPort(Class<T> serviceEndpointInterface,
                         WebServiceFeature... features) {
        return Provider.provider().getPort(this, serviceEndpointInterface,
                                           features);
    }

    /**
     * Displays EPR infoset for debugging convenience.
     */
    public String toString() {
        StringWriter w = new StringWriter();
        writeTo(new StreamResult(w));
        return w.toString();
    }
}
```

27.10 FaultAction.java

Secuencia 269: javax/xml/ws/FaultAction.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * The <code>FaultAction</code> annotation is used inside an {@link Action}
 * annotation to allow an explicit association of a WS-Addressing
 * <code>Action</code> message addressing property with the <code>fault</code>
 * messages of the WSDL operation mapped from the exception class.
 * <p>
 * The <code>wsam:Action</code> attribute value in the <code>fault</code>
 * message in the generated WSDL operation mapped for <code>className</code>
 * class is equal to the corresponding value in the <code>FaultAction</code>.
 * For the exact computation of <code>wsam:Action</code> values for the
 * fault messages, refer to the algorithm in the JAX-WS specification.
 *
 * <p>
 * <b>Example 1</b>: Specify explicit values for <code>Action</code> message addressing
 * property for the <code>input</code>, <code>output</code> and <code>fault</code> message
 * if the Java method throws only one service specific exception.
 *
 * <pre>
 * &#64;WebService(targetNamespace="http://example.com/numbers")
 * public class AddNumbersImpl {
 *     &#64;Action(
 *         fault = {
 *             <b>&#64;FaultAction(className=AddNumbersException.class, value="http://example.com/
     faultAction")</b>
 *         })
 *     public int addNumbers(int number1, int number2)
 *         throws AddNumbersException {
 *         return number1 + number2;
```

```
*     }
* }
* </pre>
*
* The generated WSDL looks like:
*
* <pre>
*   &lt;definitions targetNamespace="http://example.com/numbers" ...>
*     ...
*     &lt;portType name="AddNumbersPortType">
*       &lt;operation name="AddNumbers">
*         ...
*         &lt;fault message="tns:AddNumbersException" name="AddNumbersException"
*           <b>wsam:Action="http://example.com/faultAction"</b>/>
*       &lt;/operation>
*     &lt;/portType>
*     ...
*   &lt;/definitions>
* </pre>
*
* <p>
* Example 2: Here is an example that shows if the explicit value for <code>Action</code>
* message addressing property for the service specific exception is not present.
*
* <pre>
* &#64;WebService(targetNamespace="http://example.com/numbers")
* public class AddNumbersImpl {
*     public int addNumbers(int number1, int number2)
*         throws AddNumbersException {
*         return number1 + number2;
*     }
* }
* </pre>
*
* The generated WSDL looks like:
*
* <pre>
*   &lt;definitions targetNamespace="http://example.com/numbers" ...>
*     ...
*     &lt;portType name="AddNumbersPortType">
*       &lt;operation name="AddNumbers">
*         ...
*         &lt;fault message="tns:addNumbersFault" name="InvalidNumbers"
*           <b>wsam:Action="http://example.com/numbers/AddNumbersPortType/AddNumbers/Fault/
    AddNumbersException"</b>/>
*       &lt;/operation>
*     &lt;/portType>
*     ...
*   &lt;/definitions>
* </pre>
*
* <p>
* Example 3: Here is an example that shows how to specify explicit values for <code>Action</code>
```

```
 * message addressing property if the Java method throws more than one service specific exception.
 *
 * <pre>
 * &#64;WebService(targetNamespace="http://example.com/numbers")
 * public class AddNumbersImpl {
 *     &#64;Action(
 *         fault = {
 *             <b>&#64;FaultAction(className=AddNumbersException.class, value="http://example.com/addFaultAction"),
 *             &#64;FaultAction(className=TooBigNumbersException.class, value="http://example.com/toobigFaultAction")</b>
 *         })
 *     public int addNumbers(int number1, int number2)
 *         throws AddNumbersException, TooBigNumbersException {
 *         return number1 + number2;
 *     }
 * }
 * </pre>
 *
 * The generated WSDL looks like:
 *
 * <pre>
 *   &lt;definitions targetNamespace="http://example.com/numbers" ...>
 *     ...
 *     &lt;portType name="AddNumbersPortType">
 *       &lt;operation name="AddNumbers">
 *         ...
 *         &lt;fault message="tns:addNumbersFault" name="AddNumbersException"
 *           <b>wsam:Action="http://example.com/addFaultAction"</b>/>
 *         &lt;fault message="tns:tooBigNumbersFault" name="TooBigNumbersException"
 *           <b>wsam:Action="http://example.com/toobigFaultAction"</b>/>
 *       &lt;/operation>
 *     &lt;/portType>
 *     ...
 *   &lt;/definitions>
 * </pre>
 *
 * @since JAX-WS 2.1
 */

@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface FaultAction {
    /**
     * Name of the exception class
     */
    Class<? extends Exception> className();

    /**
     * Value of WS-Addressing <code>Action</code> message addressing property for the exception
     */
    String value() default "";
```

```
}
```

27.11 Holder.java

Secuencia 270: javax/xml/ws/Holder.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.io.Serializable;

/**
 * Holds a value of type <code>T</code>.
 *
 * @since JAX-WS 2.0
 */
public final class Holder<T> implements Serializable {

    private static final long serialVersionUID = 2623699057546497185L;

    /**
     * The value contained in the holder.
     */
    public T value;

    /**
     * Creates a new holder with a <code>null</code> value.
     */
    public Holder() {
    }
```

```java

    /**
     * Create a new holder with the specified value.
     *
     * @param value The value to be stored in the holder.
     */
    public Holder(T value) {
        this.value = value;
    }
}
```

27.12 LogicalMessage.java

Secuencia 271: javax/xml/ws/LogicalMessage.java

```java
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.transform.Source;
import javax.xml.bind.JAXBContext;

/** The <code>LogicalMessage</code> interface represents a
 *  protocol agnostic XML message and contains methods that
 *  provide access to the payload of the message.
 *
 *  @since JAX-WS 2.0
**/
public interface LogicalMessage {

  /** Gets the message payload as an XML source, may be called
```

```
   *  multiple times on the same LogicalMessage instance, always
   *  returns a new <code>Source</code> that may be used to retrieve the entire
   *  message payload.
   *
   *  <p>If the returned <code>Source</code> is an instance of
   *  <code>DOMSource</code>, then
   *  modifications to the encapsulated DOM tree change the message
   *  payload in-place, there is no need to susequently call
   *  <code>setPayload</code>. Other types of <code>Source</code> provide only
   *  read access to the message payload.
   *
   *  @return The contained message payload; returns <code>null</code> if no
   *          payload is present in this message.
  **/
  public Source getPayload();

  /** Sets the message payload
   *
   *  @param  payload message payload
   *  @throws WebServiceException If any error during the setting
   *          of the payload in this message
   *  @throws java.lang.UnsupportedOperationException If this
   *          operation is not supported
  **/
  public void setPayload(Source payload);

  /** Gets the message payload as a JAXB object. Note that there is no
   *  connection between the returned object and the message payload,
   *  changes to the payload require calling <code>setPayload</code>.
   *
   *  @param  context The JAXBContext that should be used to unmarshall
   *          the message payload
   *  @return The contained message payload; returns <code>null</code> if no
   *          payload is present in this message
   *  @throws WebServiceException If an error occurs when using a supplied
   *     JAXBContext to unmarshall the payload. The cause of
   *     the WebServiceException is the original JAXBException.
  **/
  public Object getPayload(JAXBContext context);

  /** Sets the message payload
   *
   *  @param  payload message payload
   *  @param  context The JAXBContext that should be used to marshall
   *          the payload
   *  @throws java.lang.UnsupportedOperationException If this
   *          operation is not supported
   *  @throws WebServiceException If an error occurs when using the supplied
   *     JAXBContext to marshall the payload. The cause of
   *     the WebServiceException is the original JAXBException.
  **/
  public void setPayload(Object payload, JAXBContext context);
}
```

27.13 ProtocolException.java

Secuencia 272: javax/xml/ws/ProtocolException.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

/** The <code>ProtocolException</code> class is a
 *  base class for exceptions related to a specific protocol binding. Subclasses
 *  are used to communicate protocol level fault information to clients and may
 *  be used on the server to control the protocol specific fault representation.
 *
 *  @since JAX-WS 2.0
**/
public class ProtocolException extends WebServiceException {
    /**
     * Constructs a new protocol exception with <code>null</code> as its detail message. The
     * cause is not initialized, and may subsequently be initialized by a call
     * to <code>Throwable.initCause(java.lang.Throwable)</code>.
     */
    public ProtocolException() {
        super();
    }

    /**
     * Constructs a new protocol exception with the specified detail message.
     * The cause is not initialized, and may subsequently be initialized by a
     * call to <code>Throwable.initCause(java.lang.Throwable)</code>.
     *
     * @param message the detail message. The detail message is saved for later
```

```
     *   retrieval by the Throwable.getMessage() method.
     */
    public ProtocolException(String message) {
        super(message);
    }

    /**
     * Constructs a new runtime exception with the specified detail message and
     * cause.
     *
     * Note that the detail message associated with  cause is not automatically
     * incorporated in  this runtime exception's detail message.
     *
     * @param message the detail message (which is saved for later retrieval  by
     *   the Throwable.getMessage() method).
     * @param cause the cause (which is saved for later retrieval by the
     * <code>Throwable.getCause()</code> method). (A <code>null</code> value is  permitted, and
         indicates
     * that the cause is nonexistent or  unknown.)
     */
    public ProtocolException(String message,  Throwable cause) {
        super(message, cause);
    }

    /**
     * Constructs a new runtime exception with the specified cause and a  detail
     * message of <code>(cause==null ? null : cause.toString())</code>  (which typically
     * contains the class and detail message of  cause). This constructor is
     * useful for runtime exceptions  that are little more than wrappers for
     * other throwables.
     *
     * @param cause the cause (which is saved for later retrieval by the
     * <code>Throwable.getCause()</code> method). (A <code>null</code> value is  permitted, and
         indicates
     * that the cause is nonexistent or  unknown.)
     */
    public ProtocolException(Throwable cause) {
        super(cause);
    }
}
```

27.14 Provider.java

Secuencia 273: javax/xml/ws/Provider.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

/**
 *  <p>Service endpoints may implement the <code>Provider</code>
 *  interface as a dynamic alternative to an SEI.
 *
 *  <p>Implementations are required to support <code>Provider&lt;Source&gt;</code>,
 *  <code>Provider&lt;SOAPMessage&gt;</code> and
 *  <code>Provider&lt;DataSource&gt;</code>, depending on the binding
 *  in use and the service mode.
 *
 *  <p>The <code>ServiceMode</code> annotation can be used to control whether
 *  the <code>Provider</code> instance will receive entire protocol messages
 *  or just message payloads.
 *
 *  @since JAX-WS 2.0
 *
 *  @see javax.xml.transform.Source
 *  @see javax.xml.soap.SOAPMessage
 *  @see javax.xml.ws.ServiceMode
**/
public interface Provider<T> {

  /** Invokes an operation according to the contents of the request
   *  message.
   *
   *  @param  request The request message or message payload.
   *  @return The response message or message payload. May be <code>null</code> if
             there is no response.
   *  @throws WebServiceException If there is an error processing request.
   *          The cause of the <code>WebServiceException</code> may be set to a subclass
   *          of <code>ProtocolException</code> to control the protocol level
   *          representation of the exception.
   *  @see javax.xml.ws.handler.MessageContext
   *  @see javax.xml.ws.ProtocolException
  **/
  public T invoke(T request);
```

```
}
```

27.15 RequestWrapper.java

Secuencia 274: javax/xml/ws/RequestWrapper.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;

/**
 * Used to annotate methods in the Service Endpoint Interface with the request
 * wrapper bean to be used at runtime. The default value of the <code>localName</code> is
 * the <code>operationName</code>, as defined in <code>WebMethod</code> annotation and the
 * <code>targetNamespace</code> is the target namespace of the SEI.
 * <p> When starting from Java this annotation is used resolve
 * overloading conflicts in document literal mode. Only the <code>className</code>
 * is required in this case.
 *
 *  @since JAX-WS 2.0
 **/

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
```

```java
public @interface RequestWrapper {
    /**
     * Element's local name.
     */
    public String localName() default "";

    /**
     * Element's namespace name.
     */
    public String targetNamespace() default "";

    /**
     * Request wrapper bean name.
     */
    public String className() default "";

    /**
     * wsdl:part name for the wrapper part
     *
     * @since JAX-WS 2.2
     */
    public String partName() default "";

}
```

27.16 RespectBinding.java

Secuencia 275: javax/xml/ws/RespectBinding.java

```java
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```java
package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import javax.xml.ws.spi.WebServiceFeatureAnnotation;


/**
 * This feature clarifies the use of the <code>wsdl:binding</code>
 * in a JAX-WS runtime.
 * <p>
 * This annotation MUST only be used in conjunction the
 * <code>javax.jws.WebService</code>, {@link WebServiceProvider},
 * {@link WebServiceRef} annotations.
 * When used with the <code>javax.jws.WebService</code> annotation this
 * annotation MUST only be used on the service endpoint implementation
 * class.
 * When used with a <code>WebServiceRef</code> annotation, this annotation
 * MUST only be used when a proxy instance is created. The injected SEI
 * proxy, and endpoint MUST honor the values of the <code>RespectBinding</code>
 * annotation.
 * <p>
 *
 * This annotation's behaviour is defined by the corresponding feature
 * {@link RespectBindingFeature}.
 *
 * @see RespectBindingFeature
 *
 * @since JAX-WS 2.1
 */
@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@WebServiceFeatureAnnotation(id=RespectBindingFeature.ID,bean=RespectBindingFeature.class)
public @interface RespectBinding {
    /**
     * Specifies if this feature is enabled or disabled.
     */
    boolean enabled() default true;
}
```

27.17 RespectBindingFeature.java

Secuencia 276: javax/xml/ws/RespectBindingFeature.java

```java
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.ws.soap.AddressingFeature;

/**
 * This feature clarifies the use of the <code>wsdl:binding</code>
 * in a JAX-WS runtime.
 *
 * This feature can be used during the creation of SEI proxy, and
 * {@link Dispatch} instances on the client side and {@link Endpoint}
 * instances on the server side. This feature cannot be used for {@link Service}
 * instance creation on the client side.
 * <p>
 * This feature is only useful with web services that have an
 * associated WSDL. Enabling this feature requires that a JAX-WS
 * implementation inspect the <code>wsdl:binding</code> for an
 * endpoint at runtime to make sure that all <code>wsdl:extensions</code>
 * that have the <code>required</code> attribute set to <code>true</code>
 * are understood and are being used.
 * <p>
 * The following describes the affects of this feature with respect
 * to be enabled or disabled:
 * <ul>
 *  <li> ENABLED: In this Mode, a JAX-WS runtime MUST assure that all
 *  required <code>wsdl:binding</code> extensions(including policies) are
 *  either understood and used by the runtime, or explicitly disabled by the
 *  web service application. A web service can disable a particular
 *  extension if there is a corresponding {@link WebServiceFeature} or annotation.
 *  Similarly, a web service client can disable
 *  particular extension using the corresponding <code>WebServiceFeature</code> while
 *  creating a proxy or Dispatch instance.
 *  The runtime MUST also make sure that binding of
 *  SEI parameters/return values respect the <code>wsdl:binding</code>.
 *  With this feature enabled, if a required (<code>wsdl:required="true"</code>)
```

```
 *  <code>wsdl:binding</code> extension is in the WSDL and it is not
 *  supported by a JAX-WS runtime and it has not
 *  been explicitly turned off by the web service developer, then
 *  that JAX-WS runtime MUST behave appropriately based on whether it is
 *  on the client or server:
 *  <UL>
 *    <li>Client: runtime MUST throw a
 *  {@link WebServiceException} no sooner than when one of the methods
 *  above is invoked but no later than the first invocation of an endpoint
 *  operation.
 *    <li>Server: throw a {@link WebServiceException} and the endpoint MUST fail to deploy
 *  </ul>
 *
 *  <li> DISABLED: In this Mode, an implementation may choose whether
 *  to inspect the <code>wsdl:binding</code> or not and to what degree
 *  the <code>wsdl:binding</code> will be inspected.  For example,
 *  one implementation may choose to behave as if this feature is enabled,
 *  another implementation may only choose to verify the SEI's
 *  parameter/return type bindings.
 * </ul>
 *
 * @see AddressingFeature
 *
 * @since JAX-WS 2.1
 */
public final class RespectBindingFeature extends WebServiceFeature {
    /**
     *
     * Constant value identifying the RespectBindingFeature
     */
    public static final String ID = "javax.xml.ws.RespectBindingFeature";


    /**
     * Creates an <code>RespectBindingFeature</code>.
     * The instance created will be enabled.
     */
    public RespectBindingFeature() {
        this.enabled = true;
    }

    /**
     * Creates an RespectBindingFeature
     *
     * @param enabled specifies whether this feature should
     * be enabled or not.
     */
    public RespectBindingFeature(boolean enabled) {
        this.enabled = enabled;
    }

    /**
     * {@inheritDoc}
```

```
     */
    public String getID() {
        return ID;
    }
}
```

27.18 Response.java

Secuencia 277: javax/xml/ws/Response.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.util.Map;
import java.util.concurrent.Future;

/** The <code>Response</code> interface provides methods used to obtain the
 *  payload and context of a message sent in response to an operation
 *  invocation.
 *
 *  <p>For asynchronous operation invocations it provides additional methods
 *  to check the status of the request. The <code>get(...)</code> methods may
 *  throw the standard
 *  set of exceptions and their cause may be a <code>RemoteException</code> or a
 *  {@link WebServiceException} that represents the error that occured during the
 *  asynchronous method invocation.</p>
 *
 *  @since JAX-WS 2.0
**/
public interface Response<T> extends Future<T> {
```

```
    /** Gets the contained response context.
     *
     * @return The contained response context. May be <code>null</code> if a
     * response is not yet available.
     *
    **/
    Map<String,Object> getContext();
}
```

27.19 ResponseWrapper.java

Secuencia 278: javax/xml/ws/ResponseWrapper.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;
/**
 * Used to annotate methods in the Service Endpoint Interface with the response
 * wrapper bean to be used at runtime. The default value of the <code>localName</code> is
 * the <code>operationName</code> as defined in <code>WebMethod</code> annotation appended with
 * <code>Response</code> and the <code>targetNamespace</code> is the target namespace of the SEI.
 * <p> When starting from Java this annotation is used resolve
 * overloading conflicts in document literal mode. Only the <code>className</code>
 * is required in this case.
 *
```

```
 *  @since JAX-WS 2.0
**/

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface ResponseWrapper {

    /**
     * Element's local name.
     */
    public String localName() default "";

    /**
     * Element's namespace name.
     */
    public String targetNamespace() default "";

    /**
     * Response wrapper bean name.
     */
    public String className() default "";

    /**
     * wsdl:part name for the wrapper part
     *
     * @since JAX-WS 2.2
     */
    public String partName() default "";

}
```

27.20 Service.java

Secuencia 279: javax/xml/ws/Service.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.namespace.QName;
import java.util.Iterator;
import javax.xml.ws.handler.HandlerResolver;
import javax.xml.bind.JAXBContext;
import javax.xml.ws.spi.ServiceDelegate;
import javax.xml.ws.spi.Provider;

/**
 * <code>Service</code> objects provide the client view of a Web service.
 * <p><code>Service</code> acts as a factory of the following:
 * <ul>
 * <li>Proxies for a target service endpoint.</li>
 * <li>Instances of {@link javax.xml.ws.Dispatch} for
 *     dynamic message-oriented invocation of a remote
 *     operation.
 * </li>
 * </ul>
 *
 * <p>The ports available on a service can be enumerated using the
 * <code>getPorts</code> method. Alternatively, you can pass a
 * service endpoint interface to the unary <code>getPort</code> method
 * and let the runtime select a compatible port.
 *
 * <p>Handler chains for all the objects created by a <code>Service</code>
 * can be set by means of a <code>HandlerResolver</code>.
 *
 * <p>An <code>Executor</code> may be set on the service in order
 * to gain better control over the threads used to dispatch asynchronous
 * callbacks. For instance, thread pooling with certain parameters
 * can be enabled by creating a <code>ThreadPoolExecutor</code> and
 * registering it with the service.
 *
 * @since JAX-WS 2.0
 *
 * @see javax.xml.ws.spi.Provider
 * @see javax.xml.ws.handler.HandlerResolver
 * @see java.util.concurrent.Executor
 **/
public class Service {

    private ServiceDelegate delegate;
    /**
     * The orientation of a dynamic client or service. <code>MESSAGE</code> provides
     * access to entire protocol message, <code>PAYLOAD</code> to protocol message
```

```
     * payload only.
     **/
    public enum Mode { MESSAGE, PAYLOAD }

    protected Service(java.net.URL wsdlDocumentLocation, QName serviceName) {
        delegate = Provider.provider().createServiceDelegate(wsdlDocumentLocation,
                serviceName,
                this.getClass());
    }

    protected Service(java.net.URL wsdlDocumentLocation, QName serviceName, WebServiceFeature ...
        features) {
        delegate = Provider.provider().createServiceDelegate(wsdlDocumentLocation,
                serviceName,
                this.getClass(), features);
    }


    /**
     * The <code>getPort</code> method returns a proxy. A service client
     * uses this proxy to invoke operations on the target
     * service endpoint. The <code>serviceEndpointInterface</code>
     * specifies the service endpoint interface that is supported by
     * the created dynamic proxy instance.
     *
     * @param portName  Qualified name of the service endpoint in
     *                  the WSDL service description.
     * @param serviceEndpointInterface Service endpoint interface
     *                  supported by the dynamic proxy instance.
     * @return Object Proxy instance that
     *                supports the specified service endpoint
     *                interface.
     * @throws WebServiceException This exception is thrown in the
     *                  following cases:
     *                  <UL>
     *                  <LI>If there is an error in creation of
     *                      the proxy.
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      or <code>portName</code> is specified.
     *                  </UL>
     * @see java.lang.reflect.Proxy
     * @see java.lang.reflect.InvocationHandler
     **/
    public <T> T getPort(QName portName,
            Class<T> serviceEndpointInterface) {
        return delegate.getPort(portName, serviceEndpointInterface);
    }

    /**
     * The <code>getPort</code> method returns a proxy. A service client
```

```
     * uses this proxy to invoke operations on the target
     * service endpoint. The <code>serviceEndpointInterface</code>
     * specifies the service endpoint interface that is supported by
     * the created dynamic proxy instance.
     *
     * @param portName  Qualified name of the service endpoint in
     *                  the WSDL service description.
     * @param serviceEndpointInterface Service endpoint interface
     *                  supported by the dynamic proxy instance.
     * @param features  A list of WebServiceFeatures to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that
     *                supports the specified service endpoint
     *                interface.
     * @throws WebServiceException This exception is thrown in the
     *                  following cases:
     *                  <UL>
     *                  <LI>If there is an error in creation of
     *                      the proxy.
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      or <code>portName</code> is specified.
     *                  <LI>If a feature is enabled that is not compatible
     *                      with this port or is unsupported.
     *                  </UL>
     * @see java.lang.reflect.Proxy
     * @see java.lang.reflect.InvocationHandler
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public <T> T getPort(QName portName,
            Class<T> serviceEndpointInterface, WebServiceFeature... features) {
        return delegate.getPort(portName, serviceEndpointInterface, features);
    }


    /**
     * The <code>getPort</code> method returns a proxy. The parameter
     * <code>serviceEndpointInterface</code> specifies the service
     * endpoint interface that is supported by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly.
     * The returned proxy should not be reconfigured by the client.
     *
     * @param serviceEndpointInterface Service endpoint interface.
     * @return Object instance that supports the
     *                  specified service endpoint interface.
     * @throws WebServiceException
```

```
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy.
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified.
     *                  </UL>
     **/
    public <T> T getPort(Class<T> serviceEndpointInterface) {
        return delegate.getPort(serviceEndpointInterface);
    }


    /**
     * The <code>getPort</code> method returns a proxy. The parameter
     * <code>serviceEndpointInterface</code> specifies the service
     * endpoint interface that is supported by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly.
     * The returned proxy should not be reconfigured by the client.
     *
     * @param serviceEndpointInterface Service endpoint interface.
     * @param features  A list of WebServiceFeatures to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object instance that supports the
     *                  specified service endpoint interface.
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy.
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified.
     *                  <LI>If a feature is enabled that is not compatible
     *                      with this port or is unsupported.
     *                  </UL>
     *
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public <T> T getPort(Class<T> serviceEndpointInterface,
            WebServiceFeature... features) {
        return delegate.getPort(serviceEndpointInterface, features);
    }
```

```
    /**
     * The <code>getPort</code> method returns a proxy.
     * The parameter <code>endpointReference</code> specifies the
     * endpoint that will be invoked by the returned proxy.  If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * The parameter <code>serviceEndpointInterface</code> specifies
     * the service endpoint interface that is supported by the
     * returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code> metadata
     * also has a WSDL, then the WSDL from this instance MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * The returned proxy should not be reconfigured by the client.
     * If this <code>Service</code> instance has a known proxy
     * port that matches the information contained in
     * the WSDL,
     * then that proxy is returned, otherwise a WebServiceException
     * is thrown.
     * <p>
     * Calling this method has the same behavior as the following
     * <pre>
     * <code>port = service.getPort(portName, serviceEndpointInterface);</code>
     * </pre>
     * where the <code>portName</code> is retrieved from the
     * metadata of the <code>endpointReference</code> or from the
     * <code>serviceEndpointInterface</code> and the WSDL
     * associated with this <code>Service</code> instance.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
     * returned proxy.
     * @param serviceEndpointInterface Service endpoint interface.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that supports the
     *                  specified service endpoint interface.
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy.
```

```
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If the <code>endpointReference</code> metadata does
     *                      not match the <code>serviceName</code> of this
     *                      <code>Service</code> instance.
     *                  <LI>If a <code>portName</code> cannot be extracted
     *                      from the WSDL or <code>endpointReference</code> metadata.
     *                  <LI>If an invalid
     *                      <code>endpointReference</code>
     *                      is specified.
     *                  <LI>If an invalid
     *                      <code>serviceEndpointInterface</code>
     *                      is specified.
     *                  <LI>If a feature is enabled that is not compatible
     *                      with this port or is unsupported.
     *                  </UL>
     *
     * @since JAX-WS 2.1
     **/
    public <T> T getPort(EndpointReference endpointReference,
           Class<T> serviceEndpointInterface, WebServiceFeature... features) {
        return delegate.getPort(endpointReference, serviceEndpointInterface, features);
    }

    /**
     * Creates a new port for the service. Ports created in this way contain
     * no WSDL port type information and can only be used for creating
     * <code>Dispatch</code>instances.
     *
     * @param portName  Qualified name for the target service endpoint.
     * @param bindingId A String identifier of a binding.
     * @param endpointAddress Address of the target service endpoint as a URI.
     * @throws WebServiceException If any error in the creation of
     * the port.
     *
     * @see javax.xml.ws.soap.SOAPBinding#SOAP11HTTP_BINDING
     * @see javax.xml.ws.soap.SOAPBinding#SOAP12HTTP_BINDING
     * @see javax.xml.ws.http.HTTPBinding#HTTP_BINDING
     **/
    public void addPort(QName portName, String bindingId, String endpointAddress) {
        delegate.addPort(portName, bindingId, endpointAddress);
    }


    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the client's choosing.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param type The class of object used for messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code>, <code>javax.xml.soap.SOAPMessage</code>
     * and <code>javax.activation.DataSource</code>, depending on
```

```
     * the binding in use.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body. Mode MUST be MESSAGE
     * when type is SOAPMessage.
     *
     * @return Dispatch instance.
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object.
     *
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     **/
    public <T> Dispatch<T> createDispatch(QName portName, Class<T> type, Mode mode) {
        return delegate.createDispatch(portName, type, mode);
    }


    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the client's choosing.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param type The class of object used for messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code> and <code>javax.xml.soap.SOAPMessage</code>.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body. Mode MUST be <code>MESSAGE</code>
     * when type is <code>SOAPMessage</code>.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance.
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object or if a
     *                  feature is enabled that is not compatible with
     *                  this port or is unsupported.
     *
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public <T> Dispatch<T> createDispatch(QName portName, Class<T> type,
            Service.Mode mode, WebServiceFeature... features) {
        return delegate.createDispatch(portName, type, mode, features);
```

```
    }


    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the client's choosing. If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the dispatch accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code>
     * also has a WSDL in its metadata, then the WSDL from this instance MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * An implementation MUST be able to retrieve the <code>portName</code> from the
     * <code>endpointReference</code> metadata.
     * <p>
     * This method behaves the same as calling
     * <pre>
     * <code>dispatch = service.createDispatch(portName, type, mode, features);</code>
     * </pre>
     * where the <code>portName</code> is retrieved from the
     * WSDL or <code>EndpointReference</code> metadata.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
     * returned <code>Dispatch</code> object.
     * @param type The class of object used to messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code> and <code>javax.xml.soap.SOAPMessage</code>.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body. Mode MUST be <code>MESSAGE</code>
     * when type is <code>SOAPMessage</code>.
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException
     *                  <UL>
     *                    <LI>If there is any missing WSDL metadata
```

```
     *                      as required by this method.
     *                    <li>If the <code>endpointReference</code> metadata does
     *                      not match the <code>serviceName</code> or <code>portName</code>
     *                      of a WSDL associated
     *                      with this <code>Service</code> instance.
     *                    <li>If the <code>portName</code> cannot be determined
     *                    from the <code>EndpointReference</code> metadata.
     *                    <li>If any error in the creation of
     *                     the <code>Dispatch</code> object.
     *                    <li>If a feature is enabled that is not
     *                    compatible with this port or is unsupported.
     *                  </UL>
     *
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public <T> Dispatch<T> createDispatch(EndpointReference endpointReference,
            Class<T> type, Service.Mode mode,
            WebServiceFeature... features) {
        return delegate.createDispatch(endpointReference, type, mode, features);
    }

    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body.
     *
     * @return Dispatch instance.
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object.
     *
     * @see javax.xml.bind.JAXBContext
     **/
    public Dispatch<Object> createDispatch(QName portName, JAXBContext context,
            Mode mode) {
        return delegate.createDispatch(portName, context,  mode);
    }


    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects.
```

```
     *
     * @param portName  Qualified name for the target service endpoint
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance.
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object or if a
     *                  feature is enabled that is not compatible with
     *                  this port or is unsupported.
     *
     * @see javax.xml.bind.JAXBContext
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public Dispatch<Object> createDispatch(QName portName,
            JAXBContext context, Service.Mode mode, WebServiceFeature... features) {
        return delegate.createDispatch(portName, context, mode, features);
    }


    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects. If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the dispatch accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code>
     * also has a WSDL in its metadata, then the WSDL from this instance
     * MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * An implementation MUST be able to retrieve the <code>portName</code> from the
     * <code>endpointReference</code> metadata.
```

```
     * <p>
     * This method behavies the same as calling
     * <pre>
     * <code>dispatch = service.createDispatch(portName, context, mode, features);</code>
     * </pre>
     * where the <code>portName</code> is retrieved from the
     * WSDL or <code>endpointReference</code> metadata.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
     * returned <code>Dispatch</code> object.
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the client will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the client will work with
     * SOAP messages or the contents of a SOAP body.
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException
     *                  <UL>
     *                    <li>If there is any missing WSDL metadata
     *                      as required by this method.
     *                    <li>If the <code>endpointReference</code> metadata does
     *                    not match the <code>serviceName</code> or <code>portName</code>
     *                    of a WSDL associated
     *                    with this <code>Service</code> instance.
     *                    <li>If the <code>portName</code> cannot be determined
     *                    from the <code>EndpointReference</code> metadata.
     *                    <li>If any error in the creation of
     *                    the <code>Dispatch</code> object.
     *                    <li>if a feature is enabled that is not
     *                    compatible with this port or is unsupported.
     *                  </UL>
     *
     * @see javax.xml.bind.JAXBContext
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
    **/
    public Dispatch<Object> createDispatch(EndpointReference endpointReference,
            JAXBContext context, Service.Mode mode,
            WebServiceFeature... features) {
        return delegate.createDispatch(endpointReference, context, mode, features);
    }

    /**
     * Gets the name of this service.
     * @return Qualified name of this service
```

```
     **/
    public QName getServiceName() {
        return delegate.getServiceName();
    }

    /**
     * Returns an <code>Iterator</code> for the list of
     * <code>QName</code>s of service endpoints grouped by this
     * service
     *
     * @return Returns <code>java.util.Iterator</code> with elements
     *         of type <code>javax.xml.namespace.QName</code>.
     * @throws WebServiceException If this Service class does not
     *         have access to the required WSDL metadata.
     **/
    public Iterator<javax.xml.namespace.QName> getPorts() {
        return delegate.getPorts();
    }

    /**
     * Gets the location of the WSDL document for this Service.
     *
     * @return URL for the location of the WSDL document for
     *         this service.
     **/
    public java.net.URL getWSDLDocumentLocation() {
        return delegate.getWSDLDocumentLocation();
    }

    /**
     * Returns the configured handler resolver.
     *
     * @return HandlerResolver The <code>HandlerResolver</code> being
     *         used by this <code>Service</code> instance, or <code>null</code>
     *         if there isn't one.
     **/
    public HandlerResolver getHandlerResolver() {
        return delegate.getHandlerResolver();
    }

    /**
     * Sets the <code>HandlerResolver</code> for this <code>Service</code>
     * instance.
     * <p>
     * The handler resolver, if present, will be called once for each
     * proxy or dispatch instance that is created, and the handler chain
     * returned by the resolver will be set on the instance.
     *
     * @param handlerResolver The <code>HandlerResolver</code> to use
     *        for all subsequently created proxy/dispatch objects.
     *
     * @see javax.xml.ws.handler.HandlerResolver
     **/
```

```
    public void setHandlerResolver(HandlerResolver handlerResolver) {
        delegate.setHandlerResolver(handlerResolver);
    }

    /**
     * Returns the executor for this <code>Service</code>instance.
     *
     * The executor is used for all asynchronous invocations that
     * require callbacks.
     *
     * @return The <code>java.util.concurrent.Executor</code> to be
     *         used to invoke a callback.
     *
     * @see java.util.concurrent.Executor
     **/
    public java.util.concurrent.Executor getExecutor() {
        return delegate.getExecutor();
    }

    /**
     * Sets the executor for this <code>Service</code> instance.
     *
     * The executor is used for all asynchronous invocations that
     * require callbacks.
     *
     * @param executor The <code>java.util.concurrent.Executor</code>
     *        to be used to invoke a callback.
     *
     * @throws SecurityException If the instance does not support
     *         setting an executor for security reasons (e.g. the
     *         necessary permissions are missing).
     *
     * @see java.util.concurrent.Executor
     **/
    public void setExecutor(java.util.concurrent.Executor executor) {
        delegate.setExecutor(executor);
    }

    /**
     * Creates a <code>Service</code> instance.
     *
     * The specified WSDL document location and service qualified name MUST
     * uniquely identify a <code>wsdl:service</code> element.
     *
     * @param wsdlDocumentLocation <code>URL</code> for the WSDL document location
     *                             for the service
     * @param serviceName <code>QName</code> for the service
     * @throws WebServiceException If any error in creation of the
     *                     specified service.
     **/
    public static Service create(
            java.net.URL wsdlDocumentLocation,
            QName serviceName) {
```

```
        return new Service(wsdlDocumentLocation, serviceName);
    }

    /**
     * Creates a <code>Service</code> instance. The created instance is
     * configured with the web service features.
     *
     * The specified WSDL document location and service qualified name MUST
     * uniquely identify a <code>wsdl:service</code> element.
     *
     * @param wsdlDocumentLocation <code>URL</code> for the WSDL document location
     *                             for the service
     * @param serviceName <code>QName</code> for the service
     * @param features Web Service features that must be configured on
     *        the service. If the provider doesn't understand a feature,
     *        it must throw a WebServiceException.
     * @throws WebServiceException If any error in creation of the
     *                    specified service.
     * @since JAX-WS 2.2
     **/
    public static Service create(
            java.net.URL wsdlDocumentLocation,
            QName serviceName, WebServiceFeature ... features) {
        return new Service(wsdlDocumentLocation, serviceName, features);
    }

    /**
     * Creates a <code>Service</code> instance.
     *
     * @param serviceName <code>QName</code> for the service
     * @throws WebServiceException If any error in creation of the
     *                    specified service
     */
    public static Service create(QName serviceName) {
        return new Service(null, serviceName);
    }

    /**
     * Creates a <code>Service</code> instance. The created instance is
     * configured with the web service features.
     *
     * @param serviceName <code>QName</code> for the service
     * @param features Web Service features that must be configured on
     *        the service. If the provider doesn't understand a feature,
     *        it must throw a WebServiceException.
     * @throws WebServiceException If any error in creation of the
     *                    specified service
     *
     * @since JAX-WS 2.2
     */
    public static Service create(QName serviceName, WebServiceFeature ... features) {
        return new Service(null, serviceName, features);
    }
```

```
}
```

27.21 ServiceMode.java

Secuencia 280: javax/xml/ws/ServiceMode.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Inherited;

/**
 * Used to indicate whether a {@link Provider} implementation wishes to work
 * with entire protocol messages or just with protocol message payloads.
 *
 *  @since JAX-WS 2.0
**/
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Inherited
@Documented
public @interface ServiceMode {
  /**
   * Service mode. <code>PAYLOAD</code> indicates that the <code>Provider</code> implementation
   * wishes to work with protocol message payloads only. <code>MESSAGE</code> indicates
```

```
   * that the <code>Provider</code> implementation wishes to work with entire protocol
   * messages.
  **/
  public Service.Mode value() default Service.Mode.PAYLOAD;
}
```

27.22 WebEndpoint.java

Secuencia 281: javax/xml/ws/WebEndpoint.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;

/**
 *  Used to annotate the <code>get<em>PortName</em>()</code>
 *  methods of a generated service interface.
 *
 *  <p>The information specified in this annotation is sufficient
 *  to uniquely identify a <code>wsdl:port</code> element
 *  inside a <code>wsdl:service</code>. The latter is
 *  determined based on the value of the <code>WebServiceClient</code>
 *  annotation on the generated service interface itself.
 *
 *  @since JAX-WS 2.0
```

```
 *
 *  @see javax.xml.ws.WebServiceClient
**/
@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebEndpoint {
  /**
   *  The local name of the endpoint.
  **/
  String name() default "";
}
```

27.23 WebFault.java

Secuencia 282: javax/xml/ws/WebFault.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;

/**
 * Used to annotate service specific exception classes to customize
 * to the local and namespace name of the fault element and the name
 * of the fault bean.
```

```
 *
 *  @since JAX-WS 2.0
**/
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebFault {
  /**
   *  Element's local name.
  **/
  public String name() default "";

  /**
   *  Element's namespace name.
  **/
  public String targetNamespace() default "";

  /**
   *  Fault bean name.
  **/
  public String faultBean() default "";


  /**
   *  wsdl:Message's name. Default name is the exception's class name.
   *  @since JAX-WS 2.2
   */
  public String messageName() default "";

}
```

27.24 WebServiceClient.java

Secuencia 283: javax/xml/ws/WebServiceClient.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;

/**
 *  Used to annotate a generated service interface.
 *
 *  <p>The information specified in this annotation is sufficient
 *  to uniquely identify a <code>wsdl:service</code>
 *  element inside a WSDL document. This <code>wsdl:service</code>
 *  element represents the Web service for which the generated
 *  service interface provides a client view.
 *
 *  @since JAX-WS 2.0
**/
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebServiceClient {
  /**
   *  The local name of the Web service.
  **/
  String name() default "";

  /**
   *  The namespace for the Web service.
  **/
  String targetNamespace() default "";

  /**
   *  The location of the WSDL document for the service (a URL).
  **/
  String wsdlLocation() default "";
}
```

27.25 WebServiceContext.java

Secuencia 284: javax/xml/ws/WebServiceContext.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.security.Principal;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.wsaddressing.W3CEndpointReference;
import org.w3c.dom.Element;


/**
 *  A <code>WebServiceContext</code> makes it possible for
 *  a web service endpoint implementation class to access
 *  message context and security information relative to
 *  a request being served.
 *
 *  Typically a <code>WebServiceContext</code> is injected
 *  into an endpoint implementation class using the
 *  <code>Resource</code> annotation.
 *
 *  @since JAX-WS 2.0
 *
 *  @see javax.annotation.Resource
 **/
public interface WebServiceContext {

    /**
     * Returns the <code>MessageContext</code> for the request being served
     * at the time this method is called. Only properties with
     * APPLICATION scope will be visible to the application.
     *
     * @return MessageContext The message context.
     *
     * @throws IllegalStateException This exception is thrown
     *         if the method is called while no request is
```

```
     *         being serviced.
     *
     * @see javax.xml.ws.handler.MessageContext
     * @see javax.xml.ws.handler.MessageContext.Scope
     * @see java.lang.IllegalStateException
     **/
    public MessageContext getMessageContext();

    /**
     * Returns the Principal that identifies the sender
     * of the request currently being serviced. If the
     * sender has not been authenticated, the method
     * returns <code>null</code>.
     *
     * @return Principal The principal object.
     *
     * @throws IllegalStateException This exception is thrown
     *         if the method is called while no request is
     *         being serviced.
     *
     * @see java.security.Principal
     * @see java.lang.IllegalStateException
     **/
    public Principal getUserPrincipal();

    /**
     * Returns a boolean indicating whether the
     * authenticated user is included in the specified
     * logical role. If the user has not been
     * authenticated, the method returns <code>false</code>.
     *
     * @param role  A <code>String</code> specifying the name of the role
     *
     * @return a <code>boolean</code> indicating whether
     * the sender of the request belongs to a given role
     *
     * @throws IllegalStateException This exception is thrown
     *         if the method is called while no request is
     *         being serviced.
     **/
    public boolean isUserInRole(String role);

    /**
     * Returns the <code>EndpointReference</code> for this
     * endpoint.
     * <p>
     * If the {@link Binding} for this <code>bindingProvider</code> is
     * either SOAP1.1/HTTP or SOAP1.2/HTTP, then a
     * <code>W3CEndpointReference</code> MUST be returned.
     *
     * @param referenceParameters Reference parameters to be associated with the
     * returned <code>EndpointReference</code> instance.
     * @return EndpointReference of the endpoint associated with this
```

```
     * <code>WebServiceContext</code>.
     * If the returned <code>EndpointReference</code> is of type
     * <code>W3CEndpointReference</code> then it MUST contain the
     * the specified <code>referenceParameters</code>.
     *
     * @throws IllegalStateException This exception is thrown
     *         if the method is called while no request is
     *         being serviced.
     *
     * @see W3CEndpointReference
     *
     * @since JAX-WS 2.1
     */
    public EndpointReference getEndpointReference(Element... referenceParameters);

    /**
     * Returns the <code>EndpointReference</code> associated with
     * this endpoint.
     *
     * @param clazz The type of <code>EndpointReference</code> that
     * MUST be returned.
     * @param referenceParameters Reference parameters to be associated with the
     * returned <code>EndpointReference</code> instance.
     * @return EndpointReference of type <code>clazz</code> of the endpoint
     * associated with this <code>WebServiceContext</code> instance.
     * If the returned <code>EndpointReference</code> is of type
     * <code>W3CEndpointReference</code> then it MUST contain the
     * the specified <code>referenceParameters</code>.
     *
     * @throws IllegalStateException This exception is thrown
     *         if the method is called while no request is
     *         being serviced.
     * @throws WebServiceException If the <code>clazz</code> type of
     * <code>EndpointReference</code> is not supported.
     *
     * @since JAX-WS 2.1
     **/
    public <T extends EndpointReference> T getEndpointReference(Class<T> clazz,
            Element... referenceParameters);
}
```

27.26 WebServiceException.java

Secuencia 285: javax/xml/ws/WebServiceException.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

/** The <code>WebServiceException</code> class is the base
 *  exception class for all JAX-WS API runtime exceptions.
 *
 *  @since JAX-WS 2.0
**/

public class WebServiceException extends java.lang.RuntimeException {

  /** Constructs a new exception with <code>null</code> as its
   *  detail message. The cause is not initialized.
  **/
  public WebServiceException() {
    super();
  }

  /** Constructs a new exception with the specified detail
   *  message.  The cause is not initialized.
   *  @param message The detail message which is later
   *                 retrieved using the getMessage method
  **/
  public WebServiceException(String message) {
    super(message);
  }

  /** Constructs a new exception with the specified detail
   *  message and cause.
   *
   *  @param message The detail message which is later retrieved
   *                 using the getMessage method
   *  @param cause   The cause which is saved for the later
   *                 retrieval throw by the getCause method
  **/
  public WebServiceException(String message, Throwable cause) {
    super(message,cause);
  }
```

```

  /** Constructs a new WebServiceException with the specified cause
   *  and a detail message of <tt>(cause==null ? null :
   *  cause.toString())</tt> (which typically contains the
   *  class and detail message of <tt>cause</tt>).
   *
   *  @param cause   The cause which is saved for the later
   *                 retrieval throw by the getCause method.
   *                 (A <tt>null</tt> value is permitted, and
   *                 indicates that the cause is nonexistent or
     *               unknown.)
  **/
  public WebServiceException(Throwable cause) {
    super(cause);
  }

}
```

27.27 WebServiceFeature.java

Secuencia 286: javax/xml/ws/WebServiceFeature.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;


/**
 * A WebServiceFeature is used to represent a feature that can be
 * enabled or disabled for a web service.
 * <p>
```

```
 * The JAX-WS specification will define some standard features and
 * JAX-WS implementors are free to define additional features if
 * necessary.  Vendor specific features may not be portable so
 * caution should be used when using them. Each Feature definition
 * MUST define a <code>public static final String ID</code>
 * that can be used in the Feature annotation to refer
 * to the feature. This ID MUST be unique across all features
 * of all vendors.  When defining a vendor specific feature ID,
 * use a vendor specific namespace in the ID string.
 *
 * @see javax.xml.ws.RespectBindingFeature
 * @see javax.xml.ws.soap.AddressingFeature
 * @see javax.xml.ws.soap.MTOMFeature
 *
 * @since 2.1
 */
public abstract class WebServiceFeature {
   /**
    * Each Feature definition MUST define a public static final
    * String ID that can be used in the Feature annotation to refer
    * to the feature.
    */
   // public static final String ID = "some unique feature Identifier";

   /**
    * Get the unique identifier for this WebServiceFeature.
    *
    * @return the unique identifier for this feature.
    */
   public abstract String getID();

   /**
    * Specifies if the feature is enabled or disabled
    */
   protected boolean enabled = false;


   protected WebServiceFeature(){}


   /**
    * Returns <code>true</code> if this feature is enabled.
    *
    * @return <code>true</code> if and only if the feature is enabled .
    */
   public boolean isEnabled() {
       return enabled;
   }
}
```

27.28 WebServicePermission.java

Secuencia 287: javax/xml/ws/WebServicePermission.java

```java
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.security.BasicPermission;

/**
 * This class defines web service permissions.
 * <p>
 * Web service Permissions are identified by name (also referred to as
 * a "target name") alone. There are no actions associated
 * with them.
 * <p>
 * The following permission target name is defined:
 * <p>
 * <dl>
 *   <dt>publishEndpoint
 * </dl>
 * <p>
 * The <code>publishEndpoint</code> permission allows publishing a
 * web service endpoint using the <code>publish</code> methods
 * defined by the <code>javax.xml.ws.Endpoint</code> class.
 * <p>
 * Granting <code>publishEndpoint</code> allows the application to be
 * exposed as a network service. Depending on the security of the runtime and
 * the security of the application, this may introduce a security hole that
 * is remotely exploitable.
 *
 * @see javax.xml.ws.Endpoint
 * @see java.security.BasicPermission
```

```
 * @see java.security.Permission
 * @see java.security.Permissions
 * @see java.lang.SecurityManager
 * @see java.net.SocketPermission
 */
public final class WebServicePermission extends BasicPermission {

    private static final long serialVersionUID = -146474640053770988L;

    /**
     * Creates a new permission with the specified name.
     *
     * @param name the name of the <code>WebServicePermission</code>
     */
    public WebServicePermission(String name) {
        super(name);
    }

    /**
     * Creates a new permission with the specified name and actions.
     *
     * The <code>actions</code> parameter is currently unused and
     * it should be <code>null</code>.
     *
     * @param name the name of the <code>WebServicePermission</code>
     * @param actions should be <code>null</code>
     */
    public WebServicePermission(String name, String actions) {
        super(name, actions);
    }

}
```

27.29 WebServiceProvider.java

Secuencia 288: javax/xml/ws/WebServiceProvider.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;
/**
 * Used to annotate a Provider implementation class.
 *
 * @since JAX-WS 2.0
 * @see javax.xml.ws.Provider
 */
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebServiceProvider {
    /**
     * Location of the WSDL description for the service.
     */
    String wsdlLocation() default "";

    /**
     * Service name.
     */
    String serviceName() default "";

    /**
     * Target namespace for the service
     */
    String targetNamespace() default "";

    /**
     * Port name.
     */
    String portName() default "";
}
```

27.30 WebServiceRef.java

Secuencia 289: javax/xml/ws/WebServiceRef.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import javax.xml.ws.soap.Addressing;
import javax.xml.ws.spi.WebServiceFeatureAnnotation;
import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

/**
 * The <code>WebServiceRef</code> annotation is used to
 * define a reference to a web service and
 * (optionally) an injection target for it.
 * It can be used to inject both service and proxy
 * instances. These injected references are not thread safe.
 * If the references are accessed by multiple threads,
 * usual synchronization techinques can be used to
 * support multiple threads.
 *
 * <p>
 * Web service references are resources in the Java EE 5 sense.
 * The annotations (for example, {@link Addressing}) annotated with
 * meta-annotation {@link WebServiceFeatureAnnotation}
 * can be used in conjunction with <code>WebServiceRef</code>.
 * The created reference MUST be configured with annotation's web service
 * feature.
 *
 * <p>
 * For example, in the code below, the injected
 * <code>StockQuoteProvider</code> proxy MUST
 * have WS-Addressing enabled as specifed by the
```

```
 * {@link Addressing}
 * annotation.
 *
 * <pre><code>
 *    public class MyClient {
 *       &#64;Addressing
 *       &#64;WebServiceRef(StockQuoteService.class)
 *       private StockQuoteProvider stockQuoteProvider;
 *       ...
 *    }
 * </code></pre>
 *
 * <p>
 * If a JAX-WS implementation encounters an unsupported or unrecognized
 * annotation annotated with the <code>WebServiceFeatureAnnotation</code>
 * that is specified with <code>WebServiceRef</code>, an ERROR MUST be given.
 *
 * @see javax.annotation.Resource
 * @see WebServiceFeatureAnnotation
 *
 * @since JAX-WS 2.0
 *
**/

@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebServiceRef {
    /**
     * The JNDI name of the resource.  For field annotations,
     * the default is the field name.  For method annotations,
     * the default is the JavaBeans property name corresponding
     * to the method.  For class annotations, there is no default
     * and this MUST be specified.
     *
     * The JNDI name can be absolute(with any logical namespace) or relative
     * to JNDI <code>java:comp/env</code> namespace.
     */
    String name() default "";

    /**
     * The Java type of the resource.  For field annotations,
     * the default is the type of the field.  For method annotations,
     * the default is the type of the JavaBeans property.
     * For class annotations, there is no default and this MUST be
     * specified.
     */
    Class<?> type() default Object.class;

    /**
     * A product specific name that this resource should be mapped to.
     * The name of this resource, as defined by the <code>name</code>
     * element or defaulted, is a name that is local to the application
```

```
     * component using the resource.  (When a relative JNDI name
     * is specified, then it's a name in the JNDI
     * <code>java:comp/env</code> namespace.)  Many application servers
     * provide a way to map these local names to names of resources
     * known to the application server.  This mapped name is often a
     * <i>global</i> JNDI name, but may be a name of any form.
     * <p>
     * Application servers are not required to support any particular
     * form or type of mapped name, nor the ability to use mapped names.
     * The mapped name is product-dependent and often installation-dependent.
     * No use of a mapped name is portable.
     */
    String mappedName() default "";

    /**
     * The service class, alwiays a type extending
     * <code>javax.xml.ws.Service</code>. This element MUST be specified
     * whenever the type of the reference is a service endpoint interface.
     */
    // 2.1 has Class value() default Object.class;
    // Fixing this raw Class type correctly in 2.2 API. This shouldn't cause
    // any compatibility issues for applications.
    Class<? extends Service> value() default Service.class;

    /**
     * A URL pointing to the WSDL document for the web service.
     * If not specified, the WSDL location specified by annotations
     * on the resource type is used instead.
     */
    String wsdlLocation() default "";

    /**
     * A portable JNDI lookup name that resolves to the target
     * web service reference.
     *
     * @since JAX-WS 2.2
     */
    String lookup() default "";

}
```

27.31 WebServiceRefs.java

Secuencia 290: javax/xml/ws/WebServiceRefs.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.Retention;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

/**
 * The <code>WebServiceRefs</code> annotation allows
 * multiple web service references to be declared at the
 * class level.
 *
 * <p>
 * It can be used to inject both service and proxy
 * instances. These injected references are not thread safe.
 * If the references are accessed by multiple threads,
 * usual synchronization techniques can be used to
 * support multiple threads.
 *
 * <p>
 * There is no way to associate web service features with
 * the injected instances. If an instance needs to be
 * configured with web service features, use @WebServiceRef
 * to inject the resource along with its features.
 *
 * <p>
 * <b>Example</b>: The <code>StockQuoteProvider</code>
 * proxy instance, and the <code>StockQuoteService</code> service
 * instance are injected using @WebServiceRefs.
 *
 * <pre><code>
 *    &#64;WebServiceRefs({&#64;WebServiceRef(name="service/stockquoteservice", value=
     StockQuoteService.class),
 *                     &#64;WebServiceRef(name="service/stockquoteprovider", type=
     StockQuoteProvider.class, value=StockQuoteService.class})
 *    public class MyClient {
```

```
 *        void init() {
 *            Context ic = new InitialContext();
 *            StockQuoteService service = (StockQuoteService) ic.lookup("java:comp/env/service/
     stockquoteservice");
 *            StockQuoteProvider port = (StockQuoteProvider) ic.lookup("java:comp/env/service/
     stockquoteprovider");
 *            ...
 *       }
 *       ...
 *    }
 * </code></pre>
 *
 * @see WebServiceRef
 * @since 2.0
 */

@Documented
@Retention(RUNTIME)
@Target(TYPE)
public @interface WebServiceRefs {
   /**
    * Array used for multiple web service reference declarations.
    */
   WebServiceRef[] value();
}
```

28 Xml Ws Handler (javax.xml.ws.handler package)

28.1 Handler.java

Secuencia 291: javax/xml/ws/handler/Handler.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 */

package javax.xml.ws.handler;

import javax.xml.ws.ProtocolException;
import javax.xml.ws.handler.MessageContext;

/** The <code>Handler</code> interface
 *  is the base interface for JAX-WS handlers.
 *
 *  @since JAX-WS 2.0
**/
public interface Handler<C extends MessageContext> {

  /** The <code>handleMessage</code> method is invoked for normal processing
   *  of inbound and outbound messages. Refer to the description of the handler
   *  framework in the JAX-WS specification for full details.
   *
   *  @param context the message context.
   *  @return An indication of whether handler processing should continue for
   *  the current message
   *                <ul>
   *                <li>Return <code>true</code> to continue
   *                    processing.</li>
   *                <li>Return <code>false</code> to block
   *                    processing.</li>
   *                 </ul>
   *  @throws RuntimeException Causes the JAX-WS runtime to cease
   *    handler processing and generate a fault.
   *  @throws ProtocolException Causes the JAX-WS runtime to switch to
   *    fault message processing.
  **/
  public boolean handleMessage(C context);

  /** The <code>handleFault</code> method is invoked for fault message
   *  processing.  Refer to the description of the handler
   *  framework in the JAX-WS specification for full details.
   *
   *  @param context the message context
   *  @return An indication of whether handler fault processing should continue
   *  for the current message
   *                <ul>
   *                <li>Return <code>true</code> to continue
   *                    processing.</li>
   *                <li>Return <code>false</code> to block
   *                    processing.</li>
   *                 </ul>
   *  @throws RuntimeException Causes the JAX-WS runtime to cease
   *    handler fault processing and dispatch the fault.
   *  @throws ProtocolException Causes the JAX-WS runtime to cease
   *    handler fault processing and dispatch the fault.
  **/
```

```
  public boolean handleFault(C context);

  /**
   * Called at the conclusion of a message exchange pattern just prior to
   * the JAX-WS runtime dispatching a message, fault or exception.  Refer to
   * the description of the handler
   * framework in the JAX-WS specification for full details.
   *
   * @param context the message context
  **/
  public void close(MessageContext context);
}
```

28.2 HandlerResolver.java

Secuencia 292: javax/xml/ws/handler/HandlerResolver.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler;

/**
 *  <code>HandlerResolver</code> is an interface implemented
 *  by an application to get control over the handler chain
 *  set on proxy/dispatch objects at the time of their creation.
 *  <p>
 *  A <code>HandlerResolver</code> may be set on a <code>Service</code>
 *  using the <code>setHandlerResolver</code> method.
 * <p>
 *  When the runtime invokes a <code>HandlerResolver</code>, it will
 *  pass it a <code>PortInfo</code> object containing information
```

```
 *  about the port that the proxy/dispatch object will be accessing.
 *
 *  @see javax.xml.ws.Service#setHandlerResolver
 *
 *  @since JAX-WS 2.0
**/
public interface HandlerResolver {

  /**
   *  Gets the handler chain for the specified port.
   *
   *  @param portInfo Contains information about the port being accessed.
   *  @return java.util.List&lt;Handler> chain
  **/
  public java.util.List<Handler> getHandlerChain(PortInfo portInfo);
}
```

28.3 LogicalHandler.java

Secuencia 293: javax/xml/ws/handler/LogicalHandler.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler;

/** The <code>LogicalHandler</code> extends
 *  Handler to provide typesafety for the message context parameter.
 *
 *  @since JAX-WS 2.0
**/
public interface LogicalHandler<C extends LogicalMessageContext> extends Handler<C> {
```

```
}
```

28.4 LogicalMessageContext.java

Secuencia 294: javax/xml/ws/handler/LogicalMessageContext.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler;

import javax.xml.ws.LogicalMessage;

/** The <code>LogicalMessageContext</code> interface extends
 *  <code>MessageContext</code> to
 *  provide access to a the contained message as a protocol neutral
 *  LogicalMessage
 *
 *  @since JAX-WS 2.0
**/
public interface LogicalMessageContext
                    extends MessageContext {

  /** Gets the message from this message context
   *
   *  @return The contained message; returns <code>null</code> if no
   *          message is present in this message context
  **/
  public LogicalMessage getMessage();
}
```

28.5 MessageContext.java

Secuencia 295: javax/xml/ws/handler/MessageContext.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler;
import java.util.Map;

/**
 * The interface <code>MessageContext</code> abstracts the message
 * context that is processed by a handler in the <code>handle</code>
 * method.
 *
 * <p>The <code>MessageContext</code> interface provides methods to
 * manage a property set. <code>MessageContext</code> properties
 * enable handlers in a handler chain to share processing related
 * state.
 *
 * @since JAX-WS 2.0
 */
public interface MessageContext extends Map<String, Object> {

    /**
     * Standard property: message direction, <code>true</code> for
     * outbound messages, <code>false</code> for inbound.
     * <p>Type: boolean
     */
    public static final String MESSAGE_OUTBOUND_PROPERTY =
            "javax.xml.ws.handler.message.outbound";

    /**
     * Standard property: Map of attachments to a message for the inbound
```

```
     * message, key is  the MIME Content-ID, value is a DataHandler.
     * <p>Type: java.util.Map&lt;String,DataHandler>
     */
    public static final String INBOUND_MESSAGE_ATTACHMENTS =
            "javax.xml.ws.binding.attachments.inbound";

    /**
     * Standard property: Map of attachments to a message for the outbound
     * message, key is the MIME Content-ID, value is a DataHandler.
     * <p>Type: java.util.Map&lt;String,DataHandler>
     */
    public static final String OUTBOUND_MESSAGE_ATTACHMENTS =
            "javax.xml.ws.binding.attachments.outbound";

    /**
     * Standard property: input source for WSDL document.
     * <p>Type: org.xml.sax.InputSource
     */
    public static final String WSDL_DESCRIPTION =
            "javax.xml.ws.wsdl.description";

    /**
     * Standard property: name of WSDL service.
     * <p>Type: javax.xml.namespace.QName
     */
    public static final String WSDL_SERVICE =
            "javax.xml.ws.wsdl.service";

    /**
     * Standard property: name of WSDL port.
     * <p>Type: javax.xml.namespace.QName
     */
    public static final String WSDL_PORT =
            "javax.xml.ws.wsdl.port";

    /**
     * Standard property: name of wsdl interface (2.0) or port type (1.1).
     * <p>Type: javax.xml.namespace.QName
     */
    public static final String WSDL_INTERFACE =
            "javax.xml.ws.wsdl.interface";

    /**
     * Standard property: name of WSDL operation.
     * <p>Type: javax.xml.namespace.QName
     */
    public static final String WSDL_OPERATION =
            "javax.xml.ws.wsdl.operation";

    /**
     * Standard property: HTTP response status code.
     * <p>Type: java.lang.Integer
     */
```

```
    public static final String HTTP_RESPONSE_CODE =
            "javax.xml.ws.http.response.code";

    /**
     * Standard property: HTTP request headers.
     * <p>Type: java.util.Map&lt;java.lang.String, java.util.List&lt;java.lang.String>>
     */
    public static final String HTTP_REQUEST_HEADERS =
            "javax.xml.ws.http.request.headers";

    /**
     * Standard property: HTTP response headers.
     * <p>Type: java.util.Map&lt;java.lang.String, java.util.List&lt;java.lang.String>>
     */
    public static final String HTTP_RESPONSE_HEADERS =
            "javax.xml.ws.http.response.headers";

    /**
     * Standard property: HTTP request method.
     * <p>Type: java.lang.String
     */
    public static final String HTTP_REQUEST_METHOD =
            "javax.xml.ws.http.request.method";

    /**
     * Standard property: servlet request object.
     * <p>Type: javax.servlet.http.HttpServletRequest
     */
    public static final String SERVLET_REQUEST =
            "javax.xml.ws.servlet.request";

    /**
     * Standard property: servlet response object.
     * <p>Type: javax.servlet.http.HttpServletResponse
     */
    public static final String SERVLET_RESPONSE =
            "javax.xml.ws.servlet.response";

    /**
     * Standard property: servlet context object.
     * <p>Type: javax.servlet.ServletContext
     */
    public static final String SERVLET_CONTEXT =
            "javax.xml.ws.servlet.context";

    /**
     * Standard property: Query string for request.
     * <p>Type: String
     **/
    public static final String QUERY_STRING =
            "javax.xml.ws.http.request.querystring";

    /**
```

```
     * Standard property: Request Path Info
     * <p>Type: String
     */
    public static final String PATH_INFO =
            "javax.xml.ws.http.request.pathinfo";

    /**
     * Standard property: WS Addressing Reference Parameters.
     * The list MUST include all SOAP headers marked with the
     * wsa:IsReferenceParameter="true" attribute.
     * <p>Type: List&lt;Element>
     *
     * @since JAX-WS 2.1
     */
    public static final String REFERENCE_PARAMETERS =
            "javax.xml.ws.reference.parameters";

    /**
     * Property scope. Properties scoped as <code>APPLICATION</code> are
     * visible to handlers,
     * client applications and service endpoints; properties scoped as
     * <code>HANDLER</code>
     * are only normally visible to handlers.
     */
    public enum Scope {APPLICATION, HANDLER};

    /**
     * Sets the scope of a property.
     *
     * @param name Name of the property associated with the
     *             <code>MessageContext</code>
     * @param scope Desired scope of the property
     * @throws java.lang.IllegalArgumentException if an illegal
     *             property name is specified
     */
    public void setScope(String name,  Scope scope);

    /**
     * Gets the scope of a property.
     *
     * @param name Name of the property
     * @return Scope of the property
     * @throws java.lang.IllegalArgumentException if a non-existant
     *             property name is specified
     */
    public Scope getScope(String  name);
}
```

28.6 PortInfo.java

Secuencia 296: javax/xml/ws/handler/PortInfo.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler;

import javax.xml.namespace.QName;

/**
 *  The <code>PortInfo</code> interface is used by a
 *  <code>HandlerResolver</code> to query information about
 *  the port it is being asked to create a handler chain for.
 *  <p>
 *  This interface is never implemented by an application,
 *  only by a JAX-WS implementation.
 *
 *  @since JAX-WS 2.0
**/
public interface PortInfo {

  /**
   *  Gets the qualified name of the WSDL service name containing
   *  the port being accessed.
   *
   *  @return javax.xml.namespace.QName The qualified name of the WSDL service.
  **/
  public QName getServiceName();

  /**
   *  Gets the qualified name of the WSDL port being accessed.
   *
   *  @return javax.xml.namespace.QName The qualified name of the WSDL port.
  **/
  public QName getPortName();
```

```java

  /**
   *  Gets the URI identifying the binding used by the port being accessed.
   *
   *  @return String The binding identifier for the port.
   *
   *  @see javax.xml.ws.Binding
  **/
  public String getBindingID();

}
```

29 Xml Ws Handler Soap (javax.xml.ws.handler.soap package)

29.1 SOAPHandler.java

Secuencia 297: javax/xml/ws/handler/soap/SOAPHandler.java

```java
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler.soap;

import javax.xml.namespace.QName;
import javax.xml.ws.handler.Handler;
import java.util.Set;

/** The <code>SOAPHandler</code> class extends <code>Handler</code>
 *  to provide typesafety for the message context parameter and add a method
 *  to obtain access to the headers that may be processed by the handler.
 *
 *  @since JAX-WS 2.0
```

```
**/
public interface SOAPHandler<T extends SOAPMessageContext>
    extends Handler<T> {

  /** Gets the header blocks that can be processed by this Handler
   *  instance.
   *
   *  @return Set of <code>QNames</code> of header blocks processed by this
   *          handler instance. <code>QName</code> is the qualified
   *          name of the outermost element of the Header block.
  **/
  Set<QName> getHeaders();
}
```

29.2 SOAPMessageContext.java

Secuencia 298: javax/xml/ws/handler/soap/SOAPMessageContext.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.handler.soap;

import javax.xml.soap.SOAPMessage;
import javax.xml.bind.JAXBContext;
import javax.xml.namespace.QName;
import java.util.Set;

/** The interface <code>SOAPMessageContext</code>
 *  provides access to the SOAP message for either RPC request or
 *  response. The <code>javax.xml.soap.SOAPMessage</code> specifies
 *  the standard Java API for the representation of a SOAP 1.1 message
```

```
 *  with attachments.
 *
 *  @see javax.xml.soap.SOAPMessage
 *
 *  @since JAX-WS 2.0
**/
public interface SOAPMessageContext
                    extends javax.xml.ws.handler.MessageContext {

  /** Gets the <code>SOAPMessage</code> from this message context. Modifications
   *  to the returned <code>SOAPMessage</code> change the message in-place, there
   *  is no need to subsequently call <code>setMessage</code>.
   *
   *  @return Returns the <code>SOAPMessage</code>; returns <code>null</code> if no
   *          <code>SOAPMessage</code> is present in this message context
  **/
  public SOAPMessage getMessage();

  /** Sets the SOAPMessage in this message context
   *
   *  @param  message SOAP message
   *  @throws WebServiceException If any error during the setting
   *          of the <code>SOAPMessage</code> in this message context
   *  @throws java.lang.UnsupportedOperationException If this
   *          operation is not supported
  **/
  public void setMessage(SOAPMessage message);

  /** Gets headers that have a particular qualified name from the message in the
   *  message context. Note that a SOAP message can contain multiple headers
   *  with the same qualified name.
   *
   *  @param  header The XML qualified name of the SOAP header(s).
   *  @param  context The JAXBContext that should be used to unmarshall the
   *          header
   *  @param  allRoles If <code>true</code> then returns headers for all SOAP
   *          roles, if <code>false</code> then only returns headers targetted
   *          at the roles currently being played by this SOAP node, see
   *          <code>getRoles</code>.
   *  @return An array of unmarshalled headers; returns an empty array if no
   *          message is present in this message context or no headers match
   *          the supplied qualified name.
   *  @throws WebServiceException If an error occurs when using the supplied
   *     <code>JAXBContext</code> to unmarshall. The cause of
   *     the <code>WebServiceException</code> is the original <code>JAXBException</code>.
  **/
  public Object[] getHeaders(QName header, JAXBContext context,
    boolean allRoles);

  /** Gets the SOAP actor roles associated with an execution
   *  of the handler chain.
   *  Note that SOAP actor roles apply to the SOAP node and
   *  are managed using {@link javax.xml.ws.soap.SOAPBinding#setRoles} and
```

```
   *  {@link javax.xml.ws.soap.SOAPBinding#getRoles}. <code>Handler</code> instances in
   *  the handler chain use this information about the SOAP actor
   *  roles to process the SOAP header blocks. Note that the
   *  SOAP actor roles are invariant during the processing of
   *  SOAP message through the handler chain.
   *
   *  @return Array of <code>String</code> for SOAP actor roles
  **/
  public Set<String> getRoles();
}
```

30 Xml Ws Http (javax.xml.ws.http package)

30.1 HTTPBinding.java

Secuencia 299: javax/xml/ws/http/HTTPBinding.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.http;


import javax.xml.ws.Binding;

/** The <code>HTTPBinding</code> interface is an
 *  abstraction for the XML/HTTP binding.
 *
 *  @since JAX-WS 2.0
**/
public interface HTTPBinding extends Binding {

```

```
  /**
   * A constant representing the identity of the XML/HTTP binding.
   */
  public static final String HTTP_BINDING = "http://www.w3.org/2004/08/wsdl/http";
}
```

30.2 HTTPException.java

Secuencia 300: javax/xml/ws/http/HTTPException.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.http;


/** The <code>HTTPException</code> exception represents a
 *  XML/HTTP fault.
 *
 *  <p>Since there is no standard format for faults or exceptions
 *  in XML/HTTP messaging, only the HTTP status code is captured.
 *
 *  @since JAX-WS 2.0
**/
public class HTTPException extends javax.xml.ws.ProtocolException  {

  private int statusCode;

  /** Constructor for the HTTPException
   *  @param statusCode   <code>int</code> for the HTTP status code
  **/
  public HTTPException(int statusCode) {
```

```
    super();
    this.statusCode = statusCode;
  }

  /** Gets the HTTP status code.
   *
   *  @return HTTP status code
  **/
  public int getStatusCode() {
    return statusCode;
  }
}
```

31 Xml Ws Soap (javax.xml.ws.soap package)

31.1 Addressing.java

Secuencia 301: javax/xml/ws/soap/Addressing.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.soap;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

import javax.xml.ws.BindingProvider;
import javax.xml.ws.WebServiceRef;
```

```
import javax.xml.ws.WebServiceRefs;
import javax.xml.ws.WebServiceProvider;
import javax.xml.ws.soap.AddressingFeature.Responses;
import javax.xml.ws.spi.WebServiceFeatureAnnotation;

/**
 * This annotation represents the use of WS-Addressing with either
 * the SOAP 1.1/HTTP or SOAP 1.2/HTTP binding. Using this annotation
 * with any other binding is undefined.
 * <p>
 * This annotation MUST only be used in conjunction with the
 * {@link javax.jws.WebService}, {@link WebServiceProvider},
 *  and {@link WebServiceRef} annotations.
 * When used with a <code>javax.jws.WebService</code> annotation, this
 * annotation MUST only be used on the service endpoint implementation
 * class.
 * When used with a <code>WebServiceRef</code> annotation, this annotation
 * MUST only be used when a proxy instance is created. The injected SEI
 * proxy, and endpoint MUST honor the values of the <code>Addressing</code>
 * annotation.
 * <p>
 * This annotation's behaviour is defined by the corresponding feature
 * {@link AddressingFeature}.
 *
 * @since JAX-WS 2.1
 */
@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@WebServiceFeatureAnnotation(id=AddressingFeature.ID,bean=AddressingFeature.class)
public @interface Addressing {
    /**
     * Specifies if this feature is enabled or disabled. If enabled, it means
     * the endpoint supports WS-Addressing but does not require its use.
     * Corresponding
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyaddressing">
     * 3.1.1 Addressing Assertion</a> must be generated in the generated WSDL.
     */
    boolean enabled() default true;

    /**
     * If addressing is enabled, this property determines whether the endpoint
     * requires WS-Addressing. If required is true, the endpoint requires
     * WS-Addressing and WS-Addressing headers MUST
     * be present on incoming messages. A corresponding
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyaddressing">
     * 3.1.1 Addressing Assertion</a> must be generated in the WSDL.
     */
    boolean required() default false;

    /**
     * If addressing is enabled, this property determines whether endpoint
     * requires the use of anonymous responses, or non-anonymous responses,
```

```
     * or all.
     *
     * <p>
     * {@link Responses#ALL} supports all response types and this is the
     * default value.
     *
     * <p>
     * {@link Responses#ANONYMOUS} requires the use of only anonymous
     * responses. It will result into wsam:AnonymousResponses nested assertion
     * as specified in
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyanonresponses">
     * 3.1.2 AnonymousResponses Assertion</a> in the generated WSDL.
     *
     * <p>
     * {@link Responses#NON_ANONYMOUS} requires the use of only non-anonymous
     * responses. It will result into
     * wsam:NonAnonymousResponses nested assertion as specified in
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicynonanonresponses">
     * 3.1.3 NonAnonymousResponses Assertion</a> in the generated WSDL.
     *
     * @since JAX-WS 2.2
     */
    Responses responses() default Responses.ALL;

}
```

31.2 AddressingFeature.java

Secuencia 302: javax/xml/ws/soap/AddressingFeature.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```

package javax.xml.ws.soap;

import javax.xml.ws.WebServiceFeature;
import javax.xml.ws.Endpoint;
import javax.xml.ws.Service;

/**
 * AddressingFeature represents the use of WS-Addressing with either
 * the SOAP 1.1/HTTP or SOAP 1.2/HTTP binding. Using this feature
 * with any other binding is undefined.
 * <p>
 * This feature can be used during the creation of SEI proxy, and
 * {@link javax.xml.ws.Dispatch} instances on the client side and {@link Endpoint}
 * instances on the server side. This feature cannot be used for {@link Service}
 * instance creation on the client side.
 * <p>
 * The following describes the effects of this feature with respect
 * to be enabled or disabled:
 * <ul>
 *  <li> ENABLED: In this Mode, WS-Addressing will be enabled. It means
 *       the endpoint supports WS-Addressing but does not require its use.
 *       A sender could send messages with WS-Addressing headers or without
 *       WS-Addressing headers. But a receiver MUST consume both types of
 *       messages.
 *  <li> DISABLED: In this Mode, WS-Addressing will be disabled.
 *       At runtime, WS-Addressing headers MUST NOT be used by a sender or
 *       receiver.
 * </ul>
 * <p>
 * If the feature is enabled, the <code>required</code> property determines
 * whether the endpoint requires WS-Addressing. If it is set true,
 * WS-Addressing headers MUST be present on incoming and outgoing messages.
 * By default the <code>required</code> property is <code>false</code>.
 *
 * <p>
 * If the web service developer has not explicitly enabled this feature,
 * WSDL's wsam:Addressing policy assertion is used to find
 * the use of WS-Addressing. By using the feature explicitly, an application
 * overrides WSDL's indication of the use of WS-Addressing. In some cases,
 * this is really required. For example, if an application has implemented
 * WS-Addressing itself, it can use this feature to disable addressing. That
 * means a JAX-WS implementation doesn't consume or produce WS-Addressing
 * headers.
 *
 * <p>
 * If addressing is enabled, a corresponding wsam:Addressing policy assertion
 * must be generated in the WSDL as per
 * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyassertions">
 * 3.1 WS-Policy Assertions</a>
 *
 * <p>
 * <b>Example 1: </b>Possible Policy Assertion in the generated WSDL for
```

```
 * <code>&#64;Addressing</code>
 * <pre>
 *   &lt;wsam:Addressing wsp:Optional="true">
 *     &lt;wsp:Policy/>
 *   &lt;/wsam:Addressing>
 * </pre>
 *
 * <p>
 * <b>Example 2: </b>Possible Policy Assertion in the generated WSDL for
 * <code>&#64;Addressing(required=true)</code>
 * <pre>
 *   &lt;wsam:Addressing>
 *     &lt;wsp:Policy/>
 *   &lt;/wsam:Addressing>
 * </pre>
 *
 * <p>
 * <b>Example 3: </b>Possible Policy Assertion in the generated WSDL for
 * <code>&#64;Addressing(required=true, responses=Responses.ANONYMOUS)</code>
 * <pre>
 *   &lt;wsam:Addressing>
 *      &lt;wsp:Policy>
 *        &lt;wsam:AnonymousResponses/>
 *      &lt;/wsp:Policy>
 *   &lt;/wsam:Addressing>
 * </pre>
 *
 * <p>
 * See <a href="http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/">
 * Web Services Addressing - Core</a>,
 * <a href="http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/">
 * Web Services Addressing 1.0 - SOAP Binding</a>,
 * and <a href="http://www.w3.org/TR/ws-addr-metadata/">
 * Web Services Addressing 1.0 - Metadata</a>
 * for more information on WS-Addressing.
 *
 * @see Addressing
 * @since JAX-WS 2.1
 */

public final class AddressingFeature extends WebServiceFeature {
    /**
     * Constant value identifying the AddressingFeature
     */
    public static final String ID = "http://www.w3.org/2005/08/addressing/module";

    /**
     * If addressing is enabled, this property determines whether the endpoint
     * requires WS-Addressing. If required is true, WS-Addressing headers MUST
     * be present on incoming and outgoing messages.
     */
    // should be private final, keeping original modifier due to backwards compatibility
    protected boolean required;
```

```

    /**
     * If addressing is enabled, this property determines if endpoint requires
     * the use of only anonymous responses, or only non-anonymous responses, or all.
     *
     * <p>
     * {@link Responses#ALL} supports all response types and this is the default
     * value.
     *
     * <p>
     * {@link Responses#ANONYMOUS} requires the use of only anonymous
     * responses. It will result into wsam:AnonymousResponses nested assertion
     * as specified in
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyanonresponses">
     * 3.1.2 AnonymousResponses Assertion</a> in the generated WSDL.
     *
     * <p>
     * {@link Responses#NON_ANONYMOUS} requires the use of only non-anonymous
     * responses. It will result into
     * wsam:NonAnonymousResponses nested assertion as specified in
     * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicynonanonresponses">
     * 3.1.3 NonAnonymousResponses Assertion</a> in the generated WSDL.
     *
     * @since JAX-WS 2.2
     */
    public enum Responses {
        /**
         * Specifies the use of only anonymous
         * responses. It will result into wsam:AnonymousResponses nested assertion
         * as specified in
         * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicyanonresponses">
         * 3.1.2 AnonymousResponses Assertion</a> in the generated WSDL.
         */
        ANONYMOUS,

        /**
         * Specifies the use of only non-anonymous
         * responses. It will result into
         * wsam:NonAnonymousResponses nested assertion as specified in
         * <a href="http://www.w3.org/TR/ws-addr-metadata/#wspolicynonanonresponses">
         * 3.1.3 NonAnonymousResponses Assertion</a> in the generated WSDL.
         */
        NON_ANONYMOUS,

        /**
         * Supports all response types and this is the default
         */
        ALL
    }

    private final Responses responses;

    /**
```

```
     * Creates and configures an <code>AddressingFeature</code> with the
     * use of addressing requirements. The created feature enables
     * ws-addressing i.e. supports ws-addressing but doesn't require
     * its use. It is also configured to accept all the response types.
     */
    public AddressingFeature() {
        this(true, false, Responses.ALL);
    }

    /**
     * Creates and configures an <code>AddressingFeature</code> with the
     * use of addressing requirements. If <code>enabled</code> is true,
     * it enables ws-addressing i.e. supports ws-addressing but doesn't
     * require its use. It also configures to accept all the response types.
     *
     * @param enabled true enables ws-addressing i.e.ws-addressing
     * is supported but doesn't require its use
     */
    public AddressingFeature(boolean enabled) {
        this(enabled, false, Responses.ALL);
    }

    /**
     * Creates and configures an <code>AddressingFeature</code> with the
     * use of addressing requirements. If <code>enabled</code> and
     * <code>required</code> are true, it enables ws-addressing and
     * requires its use. It also configures to accept all the response types.
     *
     * @param enabled true enables ws-addressing i.e.ws-addressing
     * is supported but doesn't require its use
     * @param required true means requires the use of ws-addressing .
     */
    public AddressingFeature(boolean enabled, boolean required) {
        this(enabled, required, Responses.ALL);
    }

    /**
     * Creates and configures an <code>AddressingFeature</code> with the
     * use of addressing requirements. If <code>enabled</code> and
     * <code>required</code> are true, it enables ws-addressing and
     * requires its use. Also, the response types can be configured using
     * <code>responses</code> parameter.
     *
     * @param enabled true enables ws-addressing i.e.ws-addressing
     * is supported but doesn't require its use
     * @param required true means requires the use of ws-addressing .
     * @param responses specifies what type of responses are required
     *
     * @since JAX-WS 2.2
     */
    public AddressingFeature(boolean enabled, boolean required, Responses responses) {
        this.enabled = enabled;
        this.required = required;
```

```
        this.responses = responses;
    }

    /**
     * {@inheritDoc}
     */
    public String getID() {
        return ID;
    }

    /**
     * If addressing is enabled, this property determines whether the endpoint
     * requires WS-Addressing. If required is true, WS-Addressing headers MUST
     * be present on incoming and outgoing messages.
     *
     * @return the current required value
     */
    public boolean isRequired() {
        return required;
    }

    /**
     * If addressing is enabled, this property determines whether endpoint
     * requires the use of anonymous responses, or non-anonymous responses,
     * or all responses.
     *
     * <p>
     * @return {@link Responses#ALL} when endpoint supports all types of
     * responses,
     *         {@link Responses#ANONYMOUS} when endpoint requires the use of
     * only anonymous responses,
     *         {@link Responses#NON_ANONYMOUS} when endpoint requires the use
     * of only non-anonymous responses
     *
     * @since JAX-WS 2.2
     */
    public Responses getResponses() {
        return responses;
    }

}
```

31.3 MTOM.java

Secuencia 303: javax/xml/ws/soap/MTOM.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.soap;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

import javax.xml.ws.spi.WebServiceFeatureAnnotation;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.WebServiceProvider;

/**
 * This feature represents the use of MTOM with a
 * web service.
 * <p>
 * This annotation MUST only be used in conjunction the
 * <code>javax.jws.WebService</code>, {@link WebServiceProvider},
 * {@link WebServiceRef} annotations.
 * When used with the <code>javax.jws.WebService</code> annotation this
 * annotation MUST only be used on the service endpoint implementation
 * class.
 * When used with a <code>WebServiceRef</code> annotation, this annotation
 * MUST only be used when a proxy instance is created. The injected SEI
 * proxy, and endpoint MUST honor the values of the <code>MTOM</code>
 * annotation.
 * <p>
 *
 * This annotation's behaviour is defined by the corresponding feature
 * {@link MTOMFeature}.
 *
 * @since JAX-WS 2.1
 */
@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
```

```
@WebServiceFeatureAnnotation(id=MTOMFeature.ID,bean=MTOMFeature.class)
public @interface MTOM {
    /**
     * Specifies if this feature is enabled or disabled.
     */
    boolean enabled() default true;

    /**
     * Property for MTOM threshold value. When MTOM is enabled, binary data above this
     * size in bytes will be XOP encoded or sent as attachment. The value of this property
     * MUST always be >= 0. Default value is 0.
     */
    int threshold() default 0;
}
```

31.4 MTOMFeature.java

Secuencia 304: javax/xml/ws/soap/MTOMFeature.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.soap;

import javax.xml.ws.WebServiceFeature;
import javax.xml.ws.WebServiceException;
import javax.xml.ws.Endpoint;
import javax.xml.ws.Service;

/**
 * This feature represents the use of MTOM with a
 * web service.
```

```
 *
 * This feature can be used during the creation of SEI proxy, and
 * {@link javax.xml.ws.Dispatch} instances on the client side and {@link Endpoint}
 * instances on the server side. This feature cannot be used for {@link Service}
 * instance creation on the client side.
 *
 * <p>
 * The following describes the affects of this feature with respect
 * to being enabled or disabled:
 * <ul>
 *  <li> ENABLED: In this Mode, MTOM will be enabled. A receiver MUST accept
 * both a non-optimized and an optimized message, and a sender MAY send an
 * optimized message, or a non-optimized message. The heuristics used by a
 * sender to determine whether to use optimization or not are
 * implementation-specific.
 *  <li> DISABLED: In this Mode, MTOM will be disabled
 * </ul>
 * <p>
 * The {@link #threshold} property can be used to set the threshold
 * value used to determine when binary data should be XOP encoded.
 *
 * @since JAX-WS 2.1
 */
public final class MTOMFeature extends WebServiceFeature {
    /**
     * Constant value identifying the MTOMFeature
     */
    public static final String ID = "http://www.w3.org/2004/08/soap/features/http-optimization";


    /**
     * Property for MTOM threshold value. This property serves as a hint when
     * MTOM is enabled, binary data above this size in bytes SHOULD be sent
     * as attachment.
     * The value of this property MUST always be >= 0. Default value is 0.
     */
    // should be changed to private final, keeping original modifier to keep backwards
        compatibility
    protected int threshold;


    /**
     * Create an <code>MTOMFeature</code>.
     * The instance created will be enabled.
     */
    public MTOMFeature() {
        this.enabled = true;
        this.threshold = 0;
    }

    /**
     * Creates an <code>MTOMFeature</code>.
     *
```

```
     * @param enabled specifies if this feature should be enabled or not
     */
    public MTOMFeature(boolean enabled) {
        this.enabled = enabled;
        this.threshold = 0;
    }


    /**
     * Creates an <code>MTOMFeature</code>.
     * The instance created will be enabled.
     *
     * @param threshold the size in bytes that binary data SHOULD be before
     * being sent as an attachment.
     *
     * @throws WebServiceException if threshold is < 0
     */
    public MTOMFeature(int threshold) {
        if (threshold < 0)
            throw new WebServiceException("MTOMFeature.threshold must be >= 0, actual value: "+
                threshold);
        this.enabled = true;
        this.threshold = threshold;
    }

    /**
     * Creates an <code>MTOMFeature</code>.
     *
     * @param enabled specifies if this feature should be enabled or not
     * @param threshold the size in bytes that binary data SHOULD be before
     * being sent as an attachment.
     *
     * @throws WebServiceException if threshold is < 0
     */
    public MTOMFeature(boolean enabled, int threshold) {
        if (threshold < 0)
            throw new WebServiceException("MTOMFeature.threshold must be >= 0, actual value: "+
                threshold);
        this.enabled = enabled;
        this.threshold = threshold;
    }

    /**
     * {@inheritDoc}
     */
    public String getID() {
        return ID;
    }

    /**
     * Gets the threshold value used to determine when binary data
     * should be sent as an attachment.
     *
```

```
     * @return the current threshold size in bytes
     */
    public int getThreshold() {
        return threshold;
    }
}
```

31.5 SOAPBinding.java

Secuencia 305: javax/xml/ws/soap/SOAPBinding.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.soap;


import java.util.Set;
import javax.xml.ws.Binding;
import javax.xml.soap.SOAPFactory;
import javax.xml.soap.MessageFactory;

/** The <code>SOAPBinding</code> interface is an abstraction for
 *  the SOAP binding.
 *
 *  @since JAX-WS 2.0
**/
public interface SOAPBinding extends Binding {

  /**
   * A constant representing the identity of the SOAP 1.1 over HTTP binding.
   */
```

```
44  public static final String SOAP11HTTP_BINDING = "http://schemas.xmlsoap.org/wsdl/soap/http";
45
46  /**
47   * A constant representing the identity of the SOAP 1.2 over HTTP binding.
48   */
49  public static final String SOAP12HTTP_BINDING = "http://www.w3.org/2003/05/soap/bindings/HTTP/";
50
51  /**
52   * A constant representing the identity of the SOAP 1.1 over HTTP binding
53   * with MTOM enabled by default.
54   */
55  public static final String SOAP11HTTP_MTOM_BINDING = "http://schemas.xmlsoap.org/wsdl/soap/http?
        mtom=true";
56
57  /**
58   * A constant representing the identity of the SOAP 1.2 over HTTP binding
59   * with MTOM enabled by default.
60   */
61  public static final String SOAP12HTTP_MTOM_BINDING = "http://www.w3.org/2003/05/soap/bindings/
        HTTP/?mtom=true";
62
63
64  /** Gets the roles played by the SOAP binding instance.
65   *
66   *  @return Set&lt;String> The set of roles played by the binding instance.
67  **/
68  public Set<String> getRoles();
69
70  /** Sets the roles played by the SOAP binding instance.
71   *
72   *  @param roles    The set of roles played by the binding instance.
73   *  @throws WebServiceException On an error in the configuration of
74   *                  the list of roles.
75  **/
76  public void setRoles(Set<String> roles);
77
78  /**
79   * Returns <code>true</code> if the use of MTOM is enabled.
80   *
81   * @return <code>true</code> if and only if the use of MTOM is enabled.
82  **/
83
84  public boolean isMTOMEnabled();
85
86  /**
87   * Enables or disables use of MTOM.
88   *
89   * @param flag   A <code>boolean</code> specifying whether the use of MTOM should
90   *               be enabled or disabled.
91   * @throws WebServiceException If the specified setting is not supported
92   *                  by this binding instance.
93   *
94   **/
```

```
  public void setMTOMEnabled(boolean flag);

  /**
   * Gets the SAAJ <code>SOAPFactory</code> instance used by this SOAP binding.
   *
   * @return SOAPFactory instance used by this SOAP binding.
  **/
  public SOAPFactory getSOAPFactory();

  /**
   * Gets the SAAJ <code>MessageFactory</code> instance used by this SOAP binding.
   *
   * @return MessageFactory instance used by this SOAP binding.
  **/
  public MessageFactory getMessageFactory();
}
```

31.6 SOAPFaultException.java

Secuencia 306: javax/xml/ws/soap/SOAPFaultException.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.soap;

import javax.xml.soap.SOAPFault;

/** The <code>SOAPFaultException</code> exception represents a
 *  SOAP 1.1 or 1.2 fault.
 *
 *  <p>A <code>SOAPFaultException</code> wraps a SAAJ <code>SOAPFault</code>
```

```
 *  that manages the SOAP-specific representation of faults.
 *  The <code>createFault</code> method of
 *  <code>javax.xml.soap.SOAPFactory</code> may be used to create an instance
 *  of <code>javax.xml.soap.SOAPFault</code> for use with the
 *  constructor. <code>SOAPBinding</code> contains an accessor for the
 *  <code>SOAPFactory</code> used by the binding instance.
 *
 *  <p>Note that the value of <code>getFault</code> is the only part of the
 *  exception used when searializing a SOAP fault.
 *
 *  <p>Refer to the SOAP specification for a complete
 *  description of SOAP faults.
 *
 *  @see javax.xml.soap.SOAPFault
 *  @see javax.xml.ws.soap.SOAPBinding#getSOAPFactory
 *  @see javax.xml.ws.ProtocolException
 *
 *  @since JAX-WS 2.0
 **/
public class SOAPFaultException extends javax.xml.ws.ProtocolException  {

    private SOAPFault fault;

    /** Constructor for SOAPFaultException
     *  @param fault   <code>SOAPFault</code> representing the fault
     *
     *  @see javax.xml.soap.SOAPFactory#createFault
     **/
    public SOAPFaultException(SOAPFault fault) {
        super(fault.getFaultString());
        this.fault = fault;
    }

    /** Gets the embedded <code>SOAPFault</code> instance.
     *
     *  @return <code>javax.xml.soap.SOAPFault</code> SOAP
     *          fault element
     **/
    public javax.xml.soap.SOAPFault getFault() {
        return this.fault;
    }
}
```

32 Xml Ws Spi (javax.xml.ws.spi package)

32.1 FactoryFinder.java

Secuencia 307: javax/xml/ws/spi/FactoryFinder.java

```
/*
 * Copyright (c) 2005, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi;

import java.io.*;

import java.util.Properties;
import javax.xml.ws.WebServiceException;

class FactoryFinder {

    /**
     * Creates an instance of the specified class using the specified
     * <code>ClassLoader</code> object.
     *
     * @exception WebServiceException if the given class could not be found
     *            or could not be instantiated
     */
    private static Object newInstance(String className,
                                      ClassLoader classLoader)
    {
        try {
            Class spiClass = safeLoadClass(className, classLoader);
            return spiClass.newInstance();
        } catch (ClassNotFoundException x) {
            throw new WebServiceException(
                "Provider " + className + " not found", x);
        } catch (Exception x) {
            throw new WebServiceException(
                "Provider " + className + " could not be instantiated: " + x,
                x);
        }
    }

    /**
```

```
     * Finds the implementation <code>Class</code> object for the given
     * factory name, or if that fails, finds the <code>Class</code> object
     * for the given fallback class name. The arguments supplied MUST be
     * used in order. If using the first argument is successful, the second
     * one will not be used.
     * <P>
     * This method is package private so that this code can be shared.
     *
     * @return the <code>Class</code> object of the specified message factory;
     *         may not be <code>null</code>
     *
     * @param factoryId             the name of the factory to find, which is
     *                              a system property
     * @param fallbackClassName     the implementation class name, which is
     *                              to be used only if nothing else
     *                              is found; <code>null</code> to indicate that
     *                              there is no fallback class name
     * @exception WebServiceException if there is an error
     */
    static Object find(String factoryId, String fallbackClassName)
    {
        if (isOsgi()) {
            return lookupUsingOSGiServiceLoader(factoryId);
        }
        ClassLoader classLoader;
        try {
            classLoader = Thread.currentThread().getContextClassLoader();
        } catch (Exception x) {
            throw new WebServiceException(x.toString(), x);
        }

        String serviceId = "META-INF/services/" + factoryId;
        // try to find services in CLASSPATH
        BufferedReader rd = null;
        try {
            InputStream is;
            if (classLoader == null) {
                is=ClassLoader.getSystemResourceAsStream(serviceId);
            } else {
                is=classLoader.getResourceAsStream(serviceId);
            }

            if( is!=null ) {
                rd = new BufferedReader(new InputStreamReader(is, "UTF-8"));

                String factoryClassName = rd.readLine();

                if (factoryClassName != null &&
                    ! "".equals(factoryClassName)) {
                    return newInstance(factoryClassName, classLoader);
                }
            }
        } catch( Exception ignored) {
```

```
        } finally {
            close(rd);
        }


        // try to read from $java.home/lib/jaxws.properties
        FileInputStream inStream = null;
        try {
            String javah=System.getProperty( "java.home" );
            String configFile = javah + File.separator +
                "lib" + File.separator + "jaxws.properties";
            File f=new File( configFile );
            if( f.exists()) {
                Properties props=new Properties();
                inStream = new FileInputStream(f);
                props.load(inStream);
                String factoryClassName = props.getProperty(factoryId);
                return newInstance(factoryClassName, classLoader);
            }
        } catch(Exception ignored) {
        } finally {
            close(inStream);
        }

        // Use the system property
        try {
            String systemProp =
                System.getProperty( factoryId );
            if( systemProp!=null) {
                return newInstance(systemProp, classLoader);
            }
        } catch (SecurityException ignored) {
        }

        if (fallbackClassName == null) {
            throw new WebServiceException(
                "Provider for " + factoryId + " cannot be found", null);
        }

        return newInstance(fallbackClassName, classLoader);
    }

    private static void close(Closeable closeable) {
        if (closeable != null) {
            try {
                closeable.close();
            } catch (IOException ignored) {
            }
        }
    }


    /**
```

```java
     * Loads the class, provided that the calling thread has an access to the class being loaded.
     */
    private static Class safeLoadClass(String className, ClassLoader classLoader) throws ClassNotFoundException {
        try {
            // make sure that the current thread has an access to the package of the given name.
            SecurityManager s = System.getSecurityManager();
            if (s != null) {
                int i = className.lastIndexOf('.');
                if (i != -1) {
                    s.checkPackageAccess(className.substring(0, i));
                }
            }

            if (classLoader == null)
                return Class.forName(className);
            else
                return classLoader.loadClass(className);
        } catch (SecurityException se) {
            // anyone can access the platform default factory class without permission
            if (Provider.DEFAULT_JAXWSPROVIDER.equals(className))
                return Class.forName(className);
            throw se;
        }
    }

    private static final String OSGI_SERVICE_LOADER_CLASS_NAME = "com.sun.org.glassfish.hk2.osgiresourcelocator.ServiceLoader";

    private static boolean isOsgi() {
        try {
            Class.forName(OSGI_SERVICE_LOADER_CLASS_NAME);
            return true;
        } catch (ClassNotFoundException ignored) {
        }
        return false;
    }

    private static Object lookupUsingOSGiServiceLoader(String factoryId) {
        try {
            // Use reflection to avoid having any dependendcy on ServiceLoader class
            Class serviceClass = Class.forName(factoryId);
            Class[] args = new Class[]{serviceClass};
            Class target = Class.forName(OSGI_SERVICE_LOADER_CLASS_NAME);
            java.lang.reflect.Method m = target.getMethod("lookupProviderInstances", Class.class);
            java.util.Iterator iter = ((Iterable) m.invoke(null, (Object[]) args)).iterator();
            return iter.hasNext() ? iter.next() : null;
        } catch (Exception ignored) {
            // log and continue
            return null;
        }
    }

```

```
}
```

32.2 Invoker.java

Secuencia 308: javax/xml/ws/spi/Invoker.java

```
/*
 * Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi;

import javax.xml.ws.WebServiceContext;
import javax.xml.ws.WebServiceFeature;
import java.lang.reflect.Method;
import java.lang.reflect.InvocationTargetException;

/**
 * Invoker hides the detail of calling into application endpoint
 * implementation. Container hands over an implementation of Invoker
 * to JAX-WS runtime, and jax-ws runtime calls {@link #invoke}
 * for a web service invocation. Finally, Invoker does the actual
 * invocation of web service on endpoint instance.
 *
 * Container also injects the provided <code>WebServiceContext</code> and takes
 * care of invoking <code>javax.annotation.PostConstruct</code> methods,
 * if present, on the endpoint implementation.
 *
 * @see Provider#createEndpoint(String, Class, Invoker, WebServiceFeature...)
 * @author Jitendra Kotamraju
 * @since JAX-WS 2.2
 */

```

```
public abstract class Invoker {

    /**
     * JAX-WS runtimes calls this method to ask container to inject
     * WebServiceContext on the endpoint instance. The
     * <code>WebServiceContext</code> object uses thread-local information
     * to return the correct information during the actual endpoint invocation
     * regardless of how many threads are concurrently being used to serve
     * requests.
     *
     * @param webServiceContext a holder for MessageContext
     * @throws IllegalAccessException if the injection done
     *         by reflection API throws this exception
     * @throws IllegalArgumentException if the injection done
     *         by reflection API throws this exception
     * @throws InvocationTargetException if the injection done
     *         by reflection API throws this exception
     */
    public abstract void inject(WebServiceContext webServiceContext)
    throws IllegalAccessException, IllegalArgumentException, InvocationTargetException;

    /**
     * JAX-WS runtime calls this method to do the actual web service
     * invocation on endpoint instance. The injected
     * <code>WebServiceContext.getMessageContext()</code> gives the correct
     * information for this invocation.
     *
     * @param m Method to be invoked on the service
     * @param args Method arguments
     * @return return value of the method
     * @throws IllegalAccessException if the invocation done
     *         by reflection API throws this exception
     * @throws IllegalArgumentException if the invocation done
     *         by reflection API throws this exception
     * @throws InvocationTargetException if the invocation done
     *         by reflection API throws this exception

     * @see Method#invoke
     */
    public abstract Object invoke(Method m, Object... args)
    throws  IllegalAccessException, IllegalArgumentException, InvocationTargetException;

}
```

32.3 Provider.java

Secuencia 309: javax/xml/ws/spi/Provider.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi;

import java.net.URL;
import java.util.List;
import java.util.Iterator;
import java.util.Map;
import java.lang.reflect.Method;
import javax.xml.namespace.QName;
import javax.xml.ws.*;
import javax.xml.ws.wsaddressing.W3CEndpointReference;

import org.w3c.dom.Element;

/**
 * Service provider for <code>ServiceDelegate</code> and
 * <code>Endpoint</code> objects.
 * <p>
 *
 * @since JAX-WS 2.0
 */
public abstract class Provider {

    /**
     * A constant representing the property used to lookup the
     * name of a <code>Provider</code> implementation
     * class.
     */
    static public final String JAXWSPROVIDER_PROPERTY
            = "javax.xml.ws.spi.Provider";

    /**
     * A constant representing the name of the default
     * <code>Provider</code> implementation class.
     **/
```

```
    // Using two strings so that package renaming doesn't change it
    static final String DEFAULT_JAXWSPROVIDER
            = "com.sun"+".xml.internal.ws.spi.ProviderImpl";

    /**
     * Take advantage of Java SE 6's java.util.ServiceLoader API.
     * Using reflection so that there is no compile-time dependency on SE 6.
     */
    static private final Method loadMethod;
    static private final Method iteratorMethod;
    static {
        Method tLoadMethod = null;
        Method tIteratorMethod = null;
        try {
            Class<?> clazz = Class.forName("java.util.ServiceLoader");
            tLoadMethod = clazz.getMethod("load", Class.class);
            tIteratorMethod = clazz.getMethod("iterator");
        } catch(ClassNotFoundException ce) {
            // Running on Java SE 5
        } catch(NoSuchMethodException ne) {
            // Shouldn't happen
        }
        loadMethod = tLoadMethod;
        iteratorMethod = tIteratorMethod;
    }


    /**
     * Creates a new instance of Provider
     */
    protected Provider() {
    }

    /**
     *
     * Creates a new provider object.
     * <p>
     * The algorithm used to locate the provider subclass to use consists
     * of the following steps:
     * <p>
     * <ul>
     * <li>
     *   If a resource with the name of
     *   <code>META-INF/services/javax.xml.ws.spi.Provider</code>
     *   exists, then its first line, if present, is used as the UTF-8 encoded
     *   name of the implementation class.
     * </li>
     * <li>
     *   If the $java.home/lib/jaxws.properties file exists and it is readable by
     *   the <code>java.util.Properties.load(InputStream)</code> method and it contains
     *   an entry whose key is <code>javax.xml.ws.spi.Provider</code>, then the value of
     *   that entry is used as the name of the implementation class.
     * </li>
```

```
     * <li>
     *   If a system property with the name <code>javax.xml.ws.spi.Provider</code>
     *   is defined, then its value is used as the name of the implementation class.
     * </li>
     * <li>
     *   Finally, a default implementation class name is used.
     * </li>
     * </ul>
     *
     */
    public static Provider provider() {
        try {
            Object provider = getProviderUsingServiceLoader();
            if (provider == null) {
                provider = FactoryFinder.find(JAXWSPROVIDER_PROPERTY, DEFAULT_JAXWSPROVIDER);
            }
            if (!(provider instanceof Provider)) {
                Class pClass = Provider.class;
                String classnameAsResource = pClass.getName().replace('.', '/') + ".class";
                ClassLoader loader = pClass.getClassLoader();
                if(loader == null) {
                    loader = ClassLoader.getSystemClassLoader();
                }
                URL targetTypeURL  = loader.getResource(classnameAsResource);
                throw new LinkageError("ClassCastException: attempting to cast" +
                       provider.getClass().getClassLoader().getResource(classnameAsResource) +
                       "to" + targetTypeURL.toString() );
            }
            return (Provider) provider;
        } catch (WebServiceException ex) {
            throw ex;
        } catch (Exception ex) {
            throw new WebServiceException("Unable to createEndpointReference Provider", ex);
        }
    }


    private static Provider getProviderUsingServiceLoader() {
        if (loadMethod != null) {
            Object loader;
            try {
                loader = loadMethod.invoke(null, Provider.class);
            } catch (Exception e) {
                throw new WebServiceException("Cannot invoke java.util.ServiceLoader#load()", e);
            }

            Iterator<Provider> it;
            try {
                it = (Iterator<Provider>)iteratorMethod.invoke(loader);
            } catch(Exception e) {
                throw new WebServiceException("Cannot invoke java.util.ServiceLoader#iterator()", e
                    );
            }
```

```
            return it.hasNext() ? it.next() : null;
        }
        return null;
    }

    /**
     * Creates a service delegate object.
     * <p>
     * @param wsdlDocumentLocation A URL pointing to the WSDL document
     *        for the service, or <code>null</code> if there isn't one.
     * @param serviceName The qualified name of the service.
     * @param serviceClass The service class, which MUST be either
     *        <code>javax.xml.ws.Service</code> or a subclass thereof.
     * @return The newly created service delegate.
     */
    public abstract ServiceDelegate createServiceDelegate(
            java.net.URL wsdlDocumentLocation,
            QName serviceName, Class<? extends Service> serviceClass);

    /**
     * Creates a service delegate object.
     * <p>
     * @param wsdlDocumentLocation A URL pointing to the WSDL document
     *        for the service, or <code>null</code> if there isn't one.
     * @param serviceName The qualified name of the service.
     * @param serviceClass The service class, which MUST be either
     *        <code>javax.xml.ws.Service</code> or a subclass thereof.
     * @param features Web Service features that must be configured on
     *        the service. If the provider doesn't understand a feature,
     *        it must throw a WebServiceException.
     * @return The newly created service delegate.
     *
     * @since JAX-WS 2.2
     */
    public ServiceDelegate createServiceDelegate(
            java.net.URL wsdlDocumentLocation,
            QName serviceName, Class<? extends Service> serviceClass, WebServiceFeature ...
                features) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }


    /**
     *
     * Creates an endpoint object with the provided binding and implementation
     * object.
     *
     * @param bindingId A URI specifying the desired binding (e.g. SOAP/HTTP)
     * @param implementor A service implementation object to which
     *        incoming requests will be dispatched. The corresponding
     *        class MUST be annotated with all the necessary Web service
     *        annotations.
```

```
     * @return The newly created endpoint.
     */
    public abstract Endpoint createEndpoint(String bindingId,
            Object implementor);


    /**
     * Creates and publishes an endpoint object with the specified
     * address and implementation object.
     *
     * @param address A URI specifying the address and transport/protocol
     *        to use. A http: URI MUST result in the SOAP 1.1/HTTP
     *        binding being used. Implementations may support other
     *        URI schemes.
     * @param implementor A service implementation object to which
     *        incoming requests will be dispatched. The corresponding
     *        class MUST be annotated with all the necessary Web service
     *        annotations.
     * @return The newly created endpoint.
     */
    public abstract Endpoint createAndPublishEndpoint(String address,
            Object implementor);

    /**
     * read an EndpointReference from the infoset contained in
     * <code>eprInfoset</code>.
     *
     * @param eprInfoset infoset for EndpointReference
     *
     * @return the <code>EndpointReference</code> unmarshalled from
     * <code>eprInfoset</code>.  This method never returns <code>null</code>.
     *
     * @throws WebServiceException If there is an error creating the
     * <code>EndpointReference</code> from the specified <code>eprInfoset</code>.
     *
     * @throws NullPointerException If the <code>null</code>
     * <code>eprInfoset</code> value is given.
     *
     * @since JAX-WS 2.1
     **/
    public abstract EndpointReference readEndpointReference(javax.xml.transform.Source eprInfoset);


    /**
     * The getPort method returns a proxy.  If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The parameter  <code>serviceEndpointInterface</code> specifies
     * the service endpoint interface that is supported by the
     * returned proxy.
     * The parameter <code>endpointReference</code> specifies the
```

```
     * endpoint that will be invoked by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly from
     * the WSDL metadata of the
     * <code>serviceEndpointInterface</code> and the <code>EndpointReference</code>.
     * For this method
     * to successfully return a proxy, WSDL metadata MUST be available and the
     * <code>endpointReference</code> MUST contain an implementation understood
     * <code>serviceName</code> metadata.
     *
     *
     * @param endpointReference the EndpointReference that will
     * be invoked by the returned proxy.
     * @param serviceEndpointInterface Service endpoint interface
     * @param features  A list of WebServiceFeatures to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that supports the
     *                  specified service endpoint interface
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method}
     *                  <LI>If this
     *                      <code>endpointReference</code>
     *                      is illegal
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified
     *                  <LI>If a feature is enabled that is not compatible with
     *                      this port or is unsupported.
     *                   </UL>
     *
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> T getPort(EndpointReference endpointReference,
            Class<T> serviceEndpointInterface,
            WebServiceFeature... features);

    /**
     * Factory method to create a <code>W3CEndpointReference</code>.
     *
     * <p>
     * This method can be used to create a <code>W3CEndpointReference</code>
     * for any endpoint by specifying the <code>address</code> property along
     * with any other desired properties.  This method
     * can also be used to create a <code>W3CEndpointReference</code> for
     * an endpoint that is published by the same Java EE application.
```

```
     * To do so the <code>address</code> property can be provided or this
     * method can automatically determine the <code>address</code> of
     * an endpoint that is published by the same Java EE application and is
     * identified by the <code>serviceName</code> and
     * <code>portName</code> propeties.  If the <code>address</code> is
     * <code>null</code> and the <code>serviceName</code> and
     * <code>portName</code> do not identify an endpoint published by the
     * same Java EE application, a
     * <code>javax.lang.IllegalStateException</code> MUST be thrown.
     *
     * @param address Specifies the address of the target endpoint
     * @param serviceName Qualified name of the service in the WSDL.
     * @param portName Qualified name of the endpoint in the WSDL.
     * @param metadata A list of elements that should be added to the
     * <code>W3CEndpointReference</code> instances <code>wsa:metadata</code>
     * element.
     * @param wsdlDocumentLocation URL for the WSDL document location for
     * the service.
     * @param referenceParameters Reference parameters to be associated
     * with the returned <code>EndpointReference</code> instance.
     *
     * @return the <code>W3CEndpointReference</code> created from
     *          <code>serviceName</code>, <code>portName</code>,
     *          <code>metadata</code>, <code>wsdlDocumentLocation</code>
     *          and <code>referenceParameters</code>. This method
     *          never returns <code>null</code>.
     *
     * @throws java.lang.IllegalStateException
     *     <ul>
     *        <li>If the <code>address</code>, <code>serviceName</code> and
     *            <code>portName</code> are all <code>null</code>.
     *        <li>If the <code>serviceName</code> service is <code>null</code> and the
     *            <code>portName</code> is NOT <code>null</code>.
     *        <li>If the <code>address</code> property is <code>null</code> and
     *            the <code>serviceName</code> and <code>portName</code> do not
     *            specify a valid endpoint published by the same Java EE
     *            application.
     *        <li>If the <code>serviceName</code>is NOT <code>null</code>
     *             and is not present in the specified WSDL.
     *        <li>If the <code>portName</code> port is not <code>null</code> and it
     *             is not present in <code>serviceName</code> service in the WSDL.
     *        <li>If the <code>wsdlDocumentLocation</code> is NOT <code>null</code>
     *            and does not represent a valid WSDL.
     *     </ul>
     * @throws WebServiceException If an error occurs while creating the
     *                             <code>W3CEndpointReference</code>.
     *
     * @since JAX-WS 2.1
     */
    public abstract W3CEndpointReference createW3CEndpointReference(String address, QName
        serviceName, QName portName,
            List<Element> metadata, String wsdlDocumentLocation, List<Element> referenceParameters)
                ;
```

```


    /**
     * Factory method to create a <code>W3CEndpointReference</code>.
     * Using this method, a <code>W3CEndpointReference</code> instance
     * can be created with extension elements, and attributes.
     * <code>Provider</code> implementations must override the default
     * implementation.
     *
     * <p>
     * This method can be used to create a <code>W3CEndpointReference</code>
     * for any endpoint by specifying the <code>address</code> property along
     * with any other desired properties.  This method
     * can also be used to create a <code>W3CEndpointReference</code> for
     * an endpoint that is published by the same Java EE application.
     * To do so the <code>address</code> property can be provided or this
     * method can automatically determine the <code>address</code> of
     * an endpoint that is published by the same Java EE application and is
     * identified by the <code>serviceName</code> and
     * <code>portName</code> propeties.  If the <code>address</code> is
     * <code>null</code> and the <code>serviceName</code> and
     * <code>portName</code> do not identify an endpoint published by the
     * same Java EE application, a
     * <code>javax.lang.IllegalStateException</code> MUST be thrown.
     *
     * @param address Specifies the address of the target endpoint
     * @param interfaceName the <code>wsam:InterfaceName</code> element in the
     * <code>wsa:Metadata</code> element.
     * @param serviceName Qualified name of the service in the WSDL.
     * @param portName Qualified name of the endpoint in the WSDL.
     * @param metadata A list of elements that should be added to the
     * <code>W3CEndpointReference</code> instances <code>wsa:metadata</code>
     * element.
     * @param wsdlDocumentLocation URL for the WSDL document location for
     * the service.
     * @param referenceParameters Reference parameters to be associated
     * with the returned <code>EndpointReference</code> instance.
     * @param elements extension elements to be associated
     * with the returned <code>EndpointReference</code> instance.
     * @param attributes extension attributes to be associated
     * with the returned <code>EndpointReference</code> instance.
     *
     * @return the <code>W3CEndpointReference</code> created from
     *          <code>serviceName</code>, <code>portName</code>,
     *          <code>metadata</code>, <code>wsdlDocumentLocation</code>
     *          and <code>referenceParameters</code>. This method
     *          never returns <code>null</code>.
     *
     * @throws java.lang.IllegalStateException
     *     <ul>
     *        <li>If the <code>address</code>, <code>serviceName</code> and
     *            <code>portName</code> are all <code>null</code>.
     *        <li>If the <code>serviceName</code> service is <code>null</code> and the
```

```
     *            <code>portName</code> is NOT <code>null</code>.
     *        <li>If the <code>address</code> property is <code>null</code> and
     *            the <code>serviceName</code> and <code>portName</code> do not
     *            specify a valid endpoint published by the same Java EE
     *            application.
     *        <li>If the <code>serviceName</code>is NOT <code>null</code>
     *             and is not present in the specified WSDL.
     *        <li>If the <code>portName</code> port is not <code>null</code> and it
     *             is not present in <code>serviceName</code> service in the WSDL.
     *        <li>If the <code>wsdlDocumentLocation</code> is NOT <code>null</code>
     *            and does not represent a valid WSDL.
     *        <li>If the <code>wsdlDocumentLocation</code> is NOT <code>null</code> but
     *            wsdli:wsdlLocation's namespace name cannot be got from the available
     *            metadata.
     *     </ul>
     * @throws WebServiceException If an error occurs while creating the
     *                             <code>W3CEndpointReference</code>.
     * @since JAX-WS 2.2
     */
    public W3CEndpointReference createW3CEndpointReference(String address,
            QName interfaceName, QName serviceName, QName portName,
            List<Element> metadata, String wsdlDocumentLocation, List<Element> referenceParameters,
            List<Element> elements, Map<QName, String> attributes) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }

    /**
     * Creates and publishes an endpoint object with the specified
     * address, implementation object and web service features.
     * <code>Provider</code> implementations must override the
     * default implementation.
     *
     * @param address A URI specifying the address and transport/protocol
     *        to use. A http: URI MUST result in the SOAP 1.1/HTTP
     *        binding being used. Implementations may support other
     *        URI schemes.
     * @param implementor A service implementation object to which
     *        incoming requests will be dispatched. The corresponding
     *        class MUST be annotated with all the necessary Web service
     *        annotations.
     * @param features A list of WebServiceFeatures to configure on the
     *        endpoint.  Supported features not in the <code>features
     *        </code> parameter will have their default values.
     * @return The newly created endpoint.
     * @since JAX-WS 2.2
     */
    public Endpoint createAndPublishEndpoint(String address,
            Object implementor, WebServiceFeature ... features) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }

```

```
    /**
     * Creates an endpoint object with the provided binding, implementation
     * object and web service features. <code>Provider</code> implementations
     * must override the default implementation.
     *
     * @param bindingId A URI specifying the desired binding (e.g. SOAP/HTTP)
     * @param implementor A service implementation object to which
     *        incoming requests will be dispatched. The corresponding
     *        class MUST be annotated with all the necessary Web service
     *        annotations.
     * @param features A list of WebServiceFeatures to configure on the
     *        endpoint.  Supported features not in the <code>features
     *        </code> parameter will have their default values.
     * @return The newly created endpoint.
     * @since JAX-WS 2.2
     */
    public Endpoint createEndpoint(String bindingId, Object implementor,
            WebServiceFeature ... features) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }

    /**
     * Creates an endpoint object with the provided binding, implementation
     * class, invoker and web service features. Containers typically use
     * this to create Endpoint objects. <code>Provider</code>
     * implementations must override the default implementation.
     *
     * @param bindingId A URI specifying the desired binding (e.g. SOAP/HTTP).
     *        Can be null.
     * @param implementorClass A service implementation class that
     *        MUST be annotated with all the necessary Web service
     *        annotations.
     * @param invoker that does the actual invocation on the service instance.
     * @param features A list of WebServiceFeatures to configure on the
     *        endpoint.  Supported features not in the <code>features
     *        </code> parameter will have their default values.
     * @return The newly created endpoint.
     * @since JAX-WS 2.2
     */
    public Endpoint createEndpoint(String bindingId, Class<?> implementorClass,
            Invoker invoker, WebServiceFeature ... features) {
        throw new UnsupportedOperationException("JAX-WS 2.2 implementation must override this
            default behaviour.");
    }

}
```

32.4 ServiceDelegate.java

Secuencia 310: javax/xml/ws/spi/ServiceDelegate.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi;

import java.util.Iterator;
import javax.xml.namespace.QName;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import javax.xml.ws.handler.HandlerResolver;
import javax.xml.ws.WebServiceFeature;
import javax.xml.bind.JAXBContext;
import javax.xml.ws.EndpointReference;
import javax.xml.ws.WebServiceException;


/**
 * Service delegates are used internally by <code>Service</code> objects
 * to allow pluggability of JAX-WS implementations.
 * <p>
 * Every <code>Service</code> object has its own delegate, created using
 * the {@link javax.xml.ws.spi.Provider#createServiceDelegate} method. A <code>Service</code>
 * object delegates all of its instance methods to its delegate.
 *
 * @see javax.xml.ws.Service
 * @see javax.xml.ws.spi.Provider
 *
 * @since JAX-WS 2.0
 */
public abstract class ServiceDelegate {

    protected ServiceDelegate() {
    }
```

```
    /**
     * The <code>getPort</code> method returns a proxy. A service client
     * uses this proxy to invoke operations on the target
     * service endpoint. The <code>serviceEndpointInterface</code>
     * specifies the service endpoint interface that is supported by
     * the created dynamic proxy instance.
     *
     * @param portName  Qualified name of the service endpoint in
     *                  the WSDL service description
     * @param serviceEndpointInterface Service endpoint interface
     *                  supported by the dynamic proxy
     * @return Object Proxy instance that
     *                supports the specified service endpoint
     *                interface
     * @throws WebServiceException This exception is thrown in the
     *                  following cases:
     *                  <UL>
     *                  <LI>If there is an error in creation of
     *                      the proxy
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      or <code>portName</code> is specified
     *                  </UL>
     * @see java.lang.reflect.Proxy
     * @see java.lang.reflect.InvocationHandler
     **/
    public abstract <T> T getPort(QName portName,
            Class<T> serviceEndpointInterface);

    /**
     * The <code>getPort</code> method returns a proxy. A service client
     * uses this proxy to invoke operations on the target
     * service endpoint. The <code>serviceEndpointInterface</code>
     * specifies the service endpoint interface that is supported by
     * the created dynamic proxy instance.
     *
     * @param portName  Qualified name of the service endpoint in
     *                  the WSDL service description
     * @param serviceEndpointInterface Service endpoint interface
     *                  supported by the dynamic proxy or instance
     * @param features  A list of WebServiceFeatures to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that
     *                supports the specified service endpoint
     *                interface
     * @throws WebServiceException This exception is thrown in the
     *                  following cases:
     *                  <UL>
     *                  <LI>If there is an error in creation of
```

```
     *                       the proxy
     *                   <LI>If there is any missing WSDL metadata
     *                       as required by this method
     *                   <LI>If an illegal
     *                       <code>serviceEndpointInterface</code>
     *                       or <code>portName</code> is specified
     *                   <LI>If a feature is enabled that is not compatible
     *                       with this port or is unsupported.
     *                   </UL>
     * @see java.lang.reflect.Proxy
     * @see java.lang.reflect.InvocationHandler
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> T getPort(QName portName,
            Class<T> serviceEndpointInterface, WebServiceFeature... features);

    /**
     * The <code>getPort</code> method returns a proxy.
     * The parameter <code>endpointReference</code> specifies the
     * endpoint that will be invoked by the returned proxy.  If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * The parameter <code>serviceEndpointInterface</code> specifies
     * the service endpoint interface that is supported by the
     * returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code> metadata
     * also has a WSDL, then the WSDL from this instance MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * The returned proxy should not be reconfigured by the client.
     * If this <code>Service</code> instance has a known proxy
     * port that matches the information contained in
     * the WSDL,
     * then that proxy is returned, otherwise a WebServiceException
     * is thrown.
     * <p>
     * Calling this method has the same behavior as the following
     * <pre>
     * <code>port = service.getPort(portName, serviceEndpointInterface);</code>
     * </pre>
```

```
     * where the <code>portName</code> is retrieved from the
     * metadata of the <code>endpointReference</code> or from the
     * <code>serviceEndpointInterface</code> and the WSDL
     * associated with this <code>Service</code> instance.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
     * returned proxy.
     * @param serviceEndpointInterface Service endpoint interface.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object Proxy instance that supports the
     *                  specified service endpoint interface.
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy.
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                  <LI>If the <code>endpointReference</code> metadata does
     *                      not match the <code>serviceName</code> of this
     *                      <code>Service</code> instance.
     *                  <LI>If a <code>portName</code> cannot be extracted
     *                      from the WSDL or <code>endpointReference</code> metadata.
     *                  <LI>If an invalid
     *                      <code>endpointReference</code>
     *                      is specified.
     *                  <LI>If an invalid
     *                      <code>serviceEndpointInterface</code>
     *                      is specified.
     *                  <LI>If a feature is enabled that is not compatible
     *                      with this port or is unsupported.
     *                  </UL>
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> T getPort(EndpointReference endpointReference,
           Class<T> serviceEndpointInterface, WebServiceFeature... features);


    /**
     * The <code>getPort</code> method returns a proxy. The parameter
     * <code>serviceEndpointInterface</code> specifies the service
     * endpoint interface that is supported by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly.
     * The returned proxy should not be reconfigured by the client.
     *
     * @param serviceEndpointInterface Service endpoint interface
     * @return Object instance that supports the
     *                  specified service endpoint interface
```

```
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified
     *                  </UL>
     **/
    public abstract <T> T getPort(Class<T> serviceEndpointInterface);


    /**
     * The <code>getPort</code> method returns a proxy. The parameter
     * <code>serviceEndpointInterface</code> specifies the service
     * endpoint interface that is supported by the returned proxy.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the proxy accordingly.
     * The returned proxy should not be reconfigured by the client.
     *
     * @param serviceEndpointInterface Service endpoint interface
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     * @return Object instance that supports the
     *                  specified service endpoint interface
     * @throws WebServiceException
     *                  <UL>
     *                  <LI>If there is an error during creation
     *                      of the proxy
     *                  <LI>If there is any missing WSDL metadata
     *                      as required by this method
     *                  <LI>If an illegal
     *                      <code>serviceEndpointInterface</code>
     *                      is specified
     *                  <LI>If a feature is enabled that is not compatible
     *                      with this port or is unsupported.
     *                  </UL>
     *
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> T getPort(Class<T> serviceEndpointInterface,
            WebServiceFeature... features);


    /**
     * Creates a new port for the service. Ports created in this way contain
     * no WSDL port type information and can only be used for creating
```

```
     * <code>Dispatch</code>instances.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param bindingId A URI identifier of a binding.
     * @param endpointAddress Address of the target service endpoint as a URI
     * @throws WebServiceException If any error in the creation of
     * the port
     *
     * @see javax.xml.ws.soap.SOAPBinding#SOAP11HTTP_BINDING
     * @see javax.xml.ws.soap.SOAPBinding#SOAP12HTTP_BINDING
     * @see javax.xml.ws.http.HTTPBinding#HTTP_BINDING
     **/
    public abstract void addPort(QName portName, String bindingId,
            String endpointAddress);




    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the user's choosing.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param type The class of object used for messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code> and <code>javax.xml.soap.SOAPMessage</code>.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the user will work with
     * SOAP messages or the contents of a SOAP body. Mode MUST be <code>MESSAGE</code>
     * when type is <code>SOAPMessage</code>.
     *
     * @return Dispatch instance
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     **/
    public abstract <T> Dispatch<T> createDispatch(QName portName, Class<T> type,
            Service.Mode mode);

    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the user's choosing.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param type The class of object used for messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code> and <code>javax.xml.soap.SOAPMessage</code>.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the user will work with
```

```
     * SOAP messages or the contents of a SOAP body. Mode MUST be <code>MESSAGE</code>
     * when type is <code>SOAPMessage</code>.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object or if a
     *                  feature is enabled that is not compatible with
     *                  this port or is unsupported.
     *
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> Dispatch<T> createDispatch(QName portName, Class<T> type,
            Service.Mode mode, WebServiceFeature... features);

    /**
     * Creates a <code>Dispatch</code> instance for use with objects of
     * the user's choosing. If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * In the implementation of this method, the JAX-WS
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the dispatch accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code>
     * also has a WSDL in its metadata, then the WSDL from this instance MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * An implementation MUST be able to retrieve the <code>portName</code> from the
     * <code>endpointReference</code> metadata.
     * <p>
     * This method behaves the same as calling
     * <pre>
     * <code>dispatch = service.createDispatch(portName, type, mode, features);</code>
     * </pre>
     * where the <code>portName</code> is retrieved from the
     * WSDL or <code>EndpointReference</code> metadata.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
```

```
     * returned <code>Dispatch</code> object.
     * @param type The class of object used to messages or message
     * payloads. Implementations are required to support
     * <code>javax.xml.transform.Source</code> and <code>javax.xml.soap.SOAPMessage</code>.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the user will work with
     * SOAP messages or the contents of a SOAP body. Mode MUST be <code>MESSAGE</code>
     * when type is <code>SOAPMessage</code>.
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException
     *                  <UL>
     *                    <LI>If there is any missing WSDL metadata
     *                      as required by this method.
     *                    <li>If the <code>endpointReference</code> metadata does
     *                      not match the <code>serviceName</code> or <code>portName</code>
     *                      of a WSDL associated
     *                      with this <code>Service</code> instance.
     *                    <li>If the <code>portName</code> cannot be determined
     *                    from the <code>EndpointReference</code> metadata.
     *                    <li>If any error in the creation of
     *                     the <code>Dispatch</code> object.
     *                    <li>If a feature is enabled that is not
     *                    compatible with this port or is unsupported.
     *                  </UL>
     *
     * @see javax.xml.transform.Source
     * @see javax.xml.soap.SOAPMessage
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract <T> Dispatch<T> createDispatch(EndpointReference endpointReference,
            Class<T> type, Service.Mode mode,
            WebServiceFeature... features);



    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
```

```
     * protocol, this parameter controls whether the user will work with
     * SOAP messages or the contents of a SOAP body.
     *
     * @return Dispatch instance
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object
     *
     * @see javax.xml.bind.JAXBContext
     **/
    public abstract Dispatch<Object> createDispatch(QName portName,
            JAXBContext context, Service.Mode mode);


    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects.
     *
     * @param portName  Qualified name for the target service endpoint
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the user will work with
     * SOAP messages or the contents of a SOAP body.
     * @param features  A list of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException If any error in the creation of
     *                  the <code>Dispatch</code> object or if a
     *                  feature is enabled that is not compatible with
     *                  this port or is unsupported.
     *
     * @see javax.xml.bind.JAXBContext
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
     **/
    public abstract Dispatch<Object> createDispatch(QName portName,
            JAXBContext context, Service.Mode mode, WebServiceFeature... features);

    /**
     * Creates a <code>Dispatch</code> instance for use with JAXB
     * generated objects. If there
     * are any reference parameters in the
     * <code>endpointReference</code>, then those reference
     * parameters MUST appear as SOAP headers, indicating them to be
     * reference parameters, on all messages sent to the endpoint.
     * The <code>endpointReference's</code> address MUST be used
     * for invocations on the endpoint.
     * In the implementation of this method, the JAX-WS
```

```
     * runtime system takes the responsibility of selecting a protocol
     * binding (and a port) and configuring the dispatch accordingly from
     * the WSDL associated with this <code>Service</code> instance or
     * from the metadata from the <code>endpointReference</code>.
     * If this <code>Service</code> instance has a WSDL and
     * the <code>endpointReference</code>
     * also has a WSDL in its metadata, then the WSDL from this instance
     * MUST be used.
     * If this <code>Service</code> instance does not have a WSDL and
     * the <code>endpointReference</code> does have a WSDL, then the
     * WSDL from the <code>endpointReference</code> MAY be used.
     * An implementation MUST be able to retrieve the <code>portName</code> from the
     * <code>endpointReference</code> metadata.
     * <p>
     * This method behavies the same as calling
     * <pre>
     * <code>dispatch = service.createDispatch(portName, context, mode, features);</code>
     * </pre>
     * where the <code>portName</code> is retrieved from the
     * WSDL or <code>endpointReference</code> metadata.
     *
     * @param endpointReference  The <code>EndpointReference</code>
     * for the target service endpoint that will be invoked by the
     * returned <code>Dispatch</code> object.
     * @param context The JAXB context used to marshall and unmarshall
     * messages or message payloads.
     * @param mode Controls whether the created dispatch instance is message
     * or payload oriented, i.e. whether the user will work with complete
     * protocol messages or message payloads. E.g. when using the SOAP
     * protocol, this parameter controls whether the user will work with
     * SOAP messages or the contents of a SOAP body.
     * @param features  An array of <code>WebServiceFeatures</code> to configure on the
     *                proxy.  Supported features not in the <code>features
     *                </code> parameter will have their default values.
     *
     * @return Dispatch instance
     * @throws WebServiceException
     *                  <UL>
     *                    <li>If there is any missing WSDL metadata
     *                      as required by this method.
     *                    <li>If the <code>endpointReference</code> metadata does
     *                    not match the <code>serviceName</code> or <code>portName</code>
     *                    of a WSDL associated
     *                    with this <code>Service</code> instance.
     *                    <li>If the <code>portName</code> cannot be determined
     *                    from the <code>EndpointReference</code> metadata.
     *                    <li>If any error in the creation of
     *                    the <code>Dispatch</code> object.
     *                    <li>if a feature is enabled that is not
     *                    compatible with this port or is unsupported.
     *                  </UL>
     *
     * @see javax.xml.bind.JAXBContext
```

```
     * @see WebServiceFeature
     *
     * @since JAX-WS 2.1
    **/
    public abstract Dispatch<Object> createDispatch(EndpointReference endpointReference,
            JAXBContext context, Service.Mode mode,
            WebServiceFeature... features);


    /**
     * Gets the name of this service.
     * @return Qualified name of this service
     **/
    public abstract QName getServiceName();

    /**
     * Returns an <code>Iterator</code> for the list of
     * <code>QName</code>s of service endpoints grouped by this
     * service
     *
     * @return Returns <code>java.util.Iterator</code> with elements
     *         of type <code>javax.xml.namespace.QName</code>
     * @throws WebServiceException If this Service class does not
     *         have access to the required WSDL metadata
     **/
    public abstract Iterator<javax.xml.namespace.QName> getPorts();

    /**
     * Gets the location of the WSDL document for this Service.
     *
     * @return URL for the location of the WSDL document for
     *         this service
     **/
    public abstract java.net.URL getWSDLDocumentLocation();

    /**
     * Returns the configured handler resolver.
     *
     * @return HandlerResolver The <code>HandlerResolver</code> being
     *         used by this <code>Service</code> instance, or <code>null</code>
     *         if there isn't one.
     **/
    public abstract HandlerResolver getHandlerResolver();

    /**
     * Sets the <code>HandlerResolver</code> for this <code>Service</code>
     * instance.
     * <p>
     * The handler resolver, if present, will be called once for each
     * proxy or dispatch instance that is created, and the handler chain
     * returned by the resolver will be set on the instance.
     *
     * @param handlerResolver The <code>HandlerResolver</code> to use
```

```
     *        for all subsequently created proxy/dispatch objects.
     *
     * @see javax.xml.ws.handler.HandlerResolver
     **/
    public abstract void setHandlerResolver(HandlerResolver handlerResolver);

    /**
     * Returns the executor for this <code>Service</code>instance.
     *
     * The executor is used for all asynchronous invocations that
     * require callbacks.
     *
     * @return The <code>java.util.concurrent.Executor</code> to be
     *         used to invoke a callback.
     *
     * @see java.util.concurrent.Executor
     **/
    public abstract java.util.concurrent.Executor getExecutor();

    /**
     * Sets the executor for this <code>Service</code> instance.
     *
     * The executor is used for all asynchronous invocations that
     * require callbacks.
     *
     * @param executor The <code>java.util.concurrent.Executor</code>
     *        to be used to invoke a callback.
     *
     * @throws SecurityException If the instance does not support
     *         setting an executor for security reasons (e.g. the
     *         necessary permissions are missing).
     *
     * @see java.util.concurrent.Executor
     **/
    public abstract void setExecutor(java.util.concurrent.Executor executor);

}
```

32.5 WebServiceFeatureAnnotation.java

Secuencia 311: javax/xml/ws/spi/WebServiceFeatureAnnotation.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi;

import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import javax.xml.ws.WebServiceFeature;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.WebServiceRefs;
import javax.xml.ws.RespectBinding;
import javax.xml.ws.soap.Addressing;
import javax.xml.ws.soap.MTOM;

/**
 * Annotation used to identify other annotations
 * as a <code>WebServiceFeature</code>.
 * <p>
 * Each <code>WebServiceFeature</code> annotation annotated with
 * this annotation MUST contain an
 * <code>enabled</code> property of type
 * <code>boolean</code> with a default value of <code>true</code>.
 * <p>
 * JAX-WS defines the following
 * <code>WebServiceFeature</code> annotations (<code>Addressing</code>,
 * <code>MTOM</code>, <code>RespectBinding</code>), however, an implementation
 * may define vendors specific annotations for other features.
 * <p>
 * Annotations annotated with <code>WebServiceFeatureAnnotation</code> MUST
 * have the same @Target of {@link WebServiceRef} annotation, so that the resulting
 * feature annotation can be used in conjunction with the {@link WebServiceRef}
 * annotation if necessary.
 * <p>
 * If a JAX-WS implementation encounters an annotation annotated
 * with the <code>WebServiceFeatureAnnotation</code> that it does not
 * recognize/support an error MUST be given.
 * <p>
 *
 * @see Addressing
 * @see MTOM
```

```
 * @see RespectBinding
 *
 * @since JAX-WS 2.1
 */
@Target(ElementType.ANNOTATION_TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebServiceFeatureAnnotation {
    /**
     * Unique identifier for the WebServiceFeature.  This
     * identifier MUST be unique across all implementations
     * of JAX-WS.
     */
    String id();

    /**
     * The <code>WebServiceFeature</code> bean that is associated
     * with the <code>WebServiceFeature</code> annotation
     */
    Class<? extends WebServiceFeature> bean();
}
```

33 Xml Ws Spi Http (javax.xml.ws.spi.http package)

33.1 HttpContext.java

Secuencia 312: javax/xml/ws/spi/http/HttpContext.java

```
/*
 * Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi.http;
```

```java

import javax.xml.ws.Endpoint;
import java.util.Set;

/**
 * HttpContext represents a mapping between the root URI path of a web
 * service to a {@link HttpHandler} which is invoked to handle requests
 * destined for that path on the associated container.
 * <p>
 * Container provides the implementation for this and it matches
 * web service requests to corresponding HttpContext objects.
 *
 * @author Jitendra Kotamraju
 * @since JAX-WS 2.2
 */
public abstract class HttpContext {

    protected HttpHandler handler;

    /**
     * JAX-WS runtime sets its handler during
     * {@link Endpoint#publish(HttpContext)} to handle
     * HTTP requests for this context. Container or its extensions
     * use this handler to process the requests.
     *
     * @param handler the handler to set for this context
     */
    public void setHandler(HttpHandler handler) {
        this.handler = handler;
    }

    /**
     * Returns the path for this context. This path uniquely identifies
     * an endpoint inside an application and the path is relative to
     * application's context path. Container should give this
     * path based on how it matches request URIs to this HttpContext object.
     *
     * <p>
     * For servlet container, this is typically a url-pattern for an endpoint.
     *
     * <p>
     * Endpoint's address for this context can be computed as follows:
     * <pre>
     *  HttpExchange exch = ...;
     *  String endpointAddress =
     *      exch.getScheme() + "://"
     *      + exch.getLocalAddress().getHostName()
     *      + ":" + exch.getLocalAddress().getPort()
     *      + exch.getContextPath() + getPath();
     * </pre>
     *
     * @return this context's path
     */
```

```
    public abstract String getPath();

    /**
     * Returns an attribute value for container's configuration
     * and other data that can be used by jax-ws runtime.
     *
     * @param name attribute name
     * @return attribute value
     */
    public abstract Object getAttribute(String name);

    /**
     * Returns all attribute names for container's configuration
     * and other data that can be used by jax-ws runtime.
     *
     * @return set of all attribute names
     */
    public abstract Set<String> getAttributeNames();

}
```

33.2 HttpExchange.java

Secuencia 313: javax/xml/ws/spi/http/HttpExchange.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi.http;

import javax.xml.ws.handler.MessageContext;
import java.io.InputStream;
```

```java
import java.io.OutputStream;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.security.Principal;

/**
 * This class encapsulates a HTTP request received and a
 * response to be generated in one exchange. It provides methods
 * for examining the request from the client, and for building and
 * sending the response.
 * <p>
 * A <code>HttpExchange</code> must be closed to free or reuse
 * underlying resources. The effect of failing to close an exchange
 * is undefined.
 *
 * @author Jitendra Kotamraju
 * @since JAX-WS 2.2
 */
public abstract class HttpExchange {

    /**
     * Standard property: cipher suite value when the request is received
     * over HTTPS
     * <p>Type: String
     */
    public static final String REQUEST_CIPHER_SUITE =
            "javax.xml.ws.spi.http.request.cipher.suite";

    /**
     * Standard property: bit size of the algorithm when the request is
     * received over HTTPS
     * <p>Type: Integer
     */
    public static final String REQUEST_KEY_SIZE =
            "javax.xml.ws.spi.http.request.key.size";

    /**
     * Standard property: A SSL certificate, if any, associated with the request
     *
     * <p>Type: java.security.cert.X509Certificate[]
     * The order of this array is defined as being in ascending order of trust.
     * The first certificate in the chain is the one set by the client, the next
     * is the one used to authenticate the first, and so on.
     */
    public static final String REQUEST_X509CERTIFICATE =
            "javax.xml.ws.spi.http.request.cert.X509Certificate";

    /**
     * Returns an immutable Map containing the HTTP headers that were
     * included with this request. The keys in this Map will be the header
```

```
     * names, while the values will be a List of Strings containing each value
     * that was included (either for a header that was listed several times,
     * or one that accepts a comma-delimited list of values on a single line).
     * In either of these cases, the values for the header name will be
     * presented in the order that they were included in the request.
     * <p>
     * The keys in Map are case-insensitive.
     *
     * @return an immutable Map which can be used to access request headers
     */
    public abstract Map<String, List<String>> getRequestHeaders();

    /**
     * Returns the value of the specified request header. If the request
     * did not include a header of the specified name, this method returns
     * null. If there are multiple headers with the same name, this method
     * returns the first header in the request. The header name is
     * case-insensitive. This is a convienence method to get a header
     * (instead of using the {@link #getRequestHeaders}).
     *
     * @param name the name of the request header
     * @return returns the value of the requested header,
     *         or null if the request does not have a header of that name
     */
     public abstract String getRequestHeader(String name);

    /**
     * Returns a mutable Map into which the HTTP response headers can be stored
     * and which will be transmitted as part of this response. The keys in the
     * Map will be the header names, while the values must be a List of Strings
     * containing each value that should be included multiple times
     * (in the order that they should be included).
     * <p>
     * The keys in Map are case-insensitive.
     *
     * @return a mutable Map which can be used to set response headers.
     */
    public abstract Map<String, List<String>> getResponseHeaders();

    /**
     * Adds a response header with the given name and value. This method
     * allows a response header to have multiple values. This is a
     * convenience method to add a response header(instead of using the
     * {@link #getResponseHeaders()}).
     *
     * @param name the name of the header
     * @param value the additional header value. If it contains octet string,
     *        it should be encoded according to
     *        RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt)
     *
     * @see #getResponseHeaders
     */
    public abstract void addResponseHeader(String name, String value);
```

```java

    /**
     * Returns the part of the request's URI from the protocol
     * name up to the query string in the first line of the HTTP request.
     * Container doesn't decode this string.
     *
     * @return the request URI
     */
    public abstract String getRequestURI();

    /**
     * Returns the context path of all the endpoints in an application.
     * This path is the portion of the request URI that indicates the
     * context of the request. The context path always comes first in a
     * request URI. The path starts with a "/" character but does not
     * end with a "/" character. If this method returns "", the request
     * is for default context. The container does not decode this string.
     *
     * <p>
     * Context path is used in computing the endpoint address. See
     * {@link HttpContext#getPath}
     *
     * @return context path of all the endpoints in an application
     * @see HttpContext#getPath
     */
    public abstract String getContextPath();

    /**
     * Get the HTTP request method
     *
     * @return the request method
     */
    public abstract String getRequestMethod();

    /**
     * Returns a {@link HttpContext} for this exchange.
     * Container matches the request with the associated Endpoint's HttpContext
     *
     * @return the HttpContext for this exchange
     */
    public abstract HttpContext getHttpContext();

    /**
     * This must be called to end an exchange. Container takes care of
     * closing request and response streams. This must be called so that
     * the container can free or reuse underlying resources.
     *
     * @throws IOException if any i/o error
     */
    public abstract void close() throws IOException;

    /**
     * Returns a stream from which the request body can be read.
```

```
     * Multiple calls to this method will return the same stream.
     *
     * @return the stream from which the request body can be read.
     * @throws IOException if any i/o error during request processing
     */
    public abstract InputStream getRequestBody() throws IOException;

    /**
     * Returns a stream to which the response body must be
     * written. {@link #setStatus}) must be called prior to calling
     * this method. Multiple calls to this method (for the same exchange)
     * will return the same stream.
     *
     * @return the stream to which the response body is written
     * @throws IOException if any i/o error during response processing
     */
    public abstract OutputStream getResponseBody() throws IOException;

    /**
     * Sets the HTTP status code for the response.
     *
     * <p>
     * This method must be called prior to calling {@link #getResponseBody}.
     *
     * @param status the response code to send
     * @see #getResponseBody
     */
    public abstract void setStatus(int status);

    /**
     * Returns the unresolved address of the remote entity invoking
     * this request.
     *
     * @return the InetSocketAddress of the caller
     */
    public abstract InetSocketAddress getRemoteAddress();

    /**
     * Returns the unresolved local address on which the request was received.
     *
     * @return the InetSocketAddress of the local interface
     */
    public abstract InetSocketAddress getLocalAddress();

    /**
     * Returns the protocol string from the request in the form
     * <i>protocol/majorVersion.minorVersion</i>. For example,
     * "HTTP/1.1"
     *
     * @return the protocol string from the request
     */
    public abstract String getProtocol();

```

```
    /**
     * Returns the name of the scheme used to make this request,
     * for example: http, or https.
     *
     * @return name of the scheme used to make this request
     */
    public abstract String getScheme();

    /**
     * Returns the extra path information that follows the web service
     * path but precedes the query string in the request URI and will start
     * with a "/" character.
     *
     * <p>
     * This can be used for {@link MessageContext#PATH_INFO}
     *
     * @return decoded extra path information of web service.
     *         It is the path that comes
     *         after the web service path but before the query string in the
     *         request URI
     *         <tt>null</tt> if there is no extra path in the request URI
     */
    public abstract String getPathInfo();

    /**
     * Returns the query string that is contained in the request URI
     * after the path.
     *
     * <p>
     * This can be used for {@link MessageContext#QUERY_STRING}
     *
     * @return undecoded query string of request URI, or
     *         <tt>null</tt> if the request URI doesn't have one
     */
    public abstract String getQueryString();

    /**
     * Returns an attribute that is associated with this
     * <code>HttpExchange</code>. JAX-WS handlers and endpoints may then
     * access the attribute via {@link MessageContext}.
     * <p>
     * Servlet containers must expose {@link MessageContext#SERVLET_CONTEXT},
     * {@link MessageContext#SERVLET_REQUEST}, and
     * {@link MessageContext#SERVLET_RESPONSE}
     * as attributes.
     *
     * <p>If the request has been received by the container using HTTPS, the
     * following information must be exposed as attributes. These attributes
     * are {@link #REQUEST_CIPHER_SUITE}, and {@link #REQUEST_KEY_SIZE}.
     * If there is a SSL certificate associated with the request, it must
     * be exposed using {@link #REQUEST_X509CERTIFICATE}
     *
     * @param name attribute name
```

```
     * @return the attribute value, or <tt>null</tt> if the attribute doesn't
     *         exist
     */
    public abstract Object getAttribute(String name);

    /**
     * Gives all the attribute names that are associated with
     * this <code>HttpExchange</code>.
     *
     * @return set of all attribute names
     * @see #getAttribute(String)
     */
    public abstract Set<String> getAttributeNames();

    /**
     * Returns the {@link Principal} that represents the authenticated
     * user for this <code>HttpExchange</code>.
     *
     * @return Principal for an authenticated user, or
     *         <tt>null</tt> if not authenticated
     */
    public abstract Principal getUserPrincipal();

    /**
     * Indicates whether an authenticated user is included in the specified
     * logical "role".
     *
     * @param role specifies the name of the role
     * @return <tt>true</tt> if the user making this request belongs to a
     *         given role
     */
    public abstract boolean isUserInRole(String role);

}
```

33.3 HttpHandler.java

Secuencia 314: javax/xml/ws/spi/http/HttpHandler.java

```
/*
 * Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.spi.http;

import javax.xml.ws.Endpoint;
import java.io.IOException;

/**
 * A handler which is invoked to process HTTP requests.
 * <p>
 * JAX-WS runtime provides the implementation for this and sets
 * it using {@link HttpContext#setHandler(HttpHandler)} during
 * {@link Endpoint#publish(HttpContext) }
 *
 * @author Jitendra Kotamraju
 * @since JAX-WS 2.2
 */
public abstract class HttpHandler {
    /**
     * Handles a given request and generates an appropriate response.
     * See {@link HttpExchange} for a description of the steps
     * involved in handling an exchange. Container invokes this method
     * when it receives an incoming request.
     *
     * @param exchange the exchange containing the request from the
     *      client and used to send the response
     * @throws IOException when an I/O error happens during request
     *      handling
     */
    public abstract void handle(HttpExchange exchange) throws IOException;
}
```

33.4 package-info.java

Secuencia 315: javax/xml/ws/spi/http/package-info.java

```
/*
 * Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/**
  Provides HTTP SPI that is used for portable deployment of JAX-WS
  web services in containers(for e.g. servlet containers). This SPI
  is not for end developers but provides a way for the container
  developers to deploy JAX-WS services portably.

  <p>
  The portable deployment is done as below:
  <ol>
  <li>Container creates {@link javax.xml.ws.Endpoint} objects for an
  application. The necessary information to create Endpoint objects
  may be got from web service deployment descriptor files.</li>
  <li>Container needs to create {@link javax.xml.ws.spi.http.HttpContext}
  objects for the deployment. For example, a HttpContext could be
  created using servlet configuration(for e.g url-pattern) for the
  web service in servlet container case.</li>
  <li>Then publishes all the endpoints using
  {@link javax.xml.ws.Endpoint#publish(HttpContext)}. During publish(),
  JAX-WS runtime registers a {@link javax.xml.ws.spi.http.HttpHandler}
  callback to handle incoming requests or
  {@link javax.xml.ws.spi.http.HttpExchange} objects. The HttpExchange
  object encapsulates a HTTP request and a response.
  </ol>

  <pre>
  Container                               JAX-WS runtime
  ---------                               --------------
  1. Creates Invoker1, ... InvokerN
  2. Provider.createEndpoint(...)     --> 3. creates Endpoint1
     configures Endpoint1
     ...
  4. Provider.createEndpoint(...)     --> 5. creates EndpointN
     configures EndpointN
  6. Creates ApplicationContext
  7. creates HttpContext1, ... HttpContextN
  8. Endpoint1.publish(HttpContext1)  --> 9. creates HttpHandler1
                                          HttpContext1.setHandler(HttpHandler1)
     ...
```

```
 10. EndpointN.publish(HttpContextN)  --> 11. creates HttpHandlerN
                                          HttpContextN.setHandler(HttpHandlerN)

  </pre>

  The request processing is done as below(for every request):
  <pre>
  Container                                JAX-WS runtime
  ---------                                --------------
  1. Creates a HttpExchange
  2. Gets handler from HttpContext
  3. HttpHandler.handle(HttpExchange) --> 4. reads request from HttpExchange
                                      <-- 5. Calls Invoker
  6. Invokes the actual instance
                                           7. Writes the response to HttpExchange
  </pre>

  <p>
  The portable undeployment is done as below:
  <pre>
  Container
  ---------
  1. @preDestroy on instances
  2. Endpoint1.stop()
  ...
  3. EndpointN.stop()
  </pre>

  @author Jitendra Kotamraju
  @since JAX-WS 2.2
 */
package javax.xml.ws.spi.http;
```

34 Xml Ws Wsaddressing (javax.xml.ws.wsaddressing package)

34.1 W3CEndpointReference.java

Secuencia 316: javax/xml/ws/wsaddressing/W3CEndpointReference.java

```
/*
 * Copyright (c) 2005, 2014, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.wsaddressing;


import org.w3c.dom.Element;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.annotation.XmlAnyAttribute;
import javax.xml.bind.annotation.XmlAnyElement;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.XmlValue;
import javax.xml.namespace.QName;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.ws.EndpointReference;
import javax.xml.ws.WebServiceException;
import java.util.List;
import java.util.Map;


/**
 * This class represents a W3C Addressing EndpointReferece which is
 * a remote reference to a web service endpoint that supports the
 * W3C WS-Addressing 1.0 - Core Recommendation.
 * <p>
 * Developers should use this class in their SEIs if they want to
 * pass/return endpoint references that represent the W3C WS-Addressing
 * recommendation.
 * <p>
 * JAXB will use the JAXB annotations and bind this class to XML infoset
 * that is consistent with that defined by WS-Addressing.  See
 * <a href="http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/">
 * WS-Addressing</a>
 * for more information on WS-Addressing EndpointReferences.
 *
 * @since JAX-WS 2.1
 */

// XmlRootElement allows this class to be marshalled on its own
@XmlRootElement(name="EndpointReference",namespace=W3CEndpointReference.NS)
```

```java
@XmlType(name="EndpointReferenceType",namespace=W3CEndpointReference.NS)
public final class W3CEndpointReference extends EndpointReference {

    private final JAXBContext w3cjc = getW3CJaxbContext();

    // should be changed to package private, keeping original modifier to keep backwards compatibility
    protected static final String NS = "http://www.w3.org/2005/08/addressing";

    // default constructor forbidden ...
    // should be private, keeping original modifier to keep backwards compatibility
    protected W3CEndpointReference() {
    }

    /**
     * Creates an EPR from infoset representation
     *
     * @param source A source object containing valid XmlInfoset
     * instance consistent with the W3C WS-Addressing Core
     * recommendation.
     *
     * @throws WebServiceException
     *   If the source does NOT contain a valid W3C WS-Addressing
     *   EndpointReference.
     * @throws NullPointerException
     *   If the <code>null</code> <code>source</code> value is given
     */
    public W3CEndpointReference(Source source) {
        try {
            W3CEndpointReference epr = w3cjc.createUnmarshaller().unmarshal(source, W3CEndpointReference.class).getValue();
            this.address = epr.address;
            this.metadata = epr.metadata;
            this.referenceParameters = epr.referenceParameters;
            this.elements = epr.elements;
            this.attributes = epr.attributes;
        } catch (JAXBException e) {
            throw new WebServiceException("Error unmarshalling W3CEndpointReference " ,e);
        } catch (ClassCastException e) {
            throw new WebServiceException("Source did not contain W3CEndpointReference", e);
        }
    }

    /**
     * {@inheritDoc}
     */
    public void writeTo(Result result){
        try {
            Marshaller marshaller = w3cjc.createMarshaller();
            marshaller.marshal(this, result);
        } catch (JAXBException e) {
            throw new WebServiceException("Error marshalling W3CEndpointReference. ", e);
        }
```

```
    }

    private static JAXBContext getW3CJaxbContext() {
        try {
            return JAXBContext.newInstance(W3CEndpointReference.class);
        } catch (JAXBException e) {
            throw new WebServiceException("Error creating JAXBContext for W3CEndpointReference. ",
                e);
        }
    }

    // private but necessary properties for databinding
    @XmlElement(name="Address",namespace=NS)
    private Address address;
    @XmlElement(name="ReferenceParameters",namespace=NS)
    private Elements referenceParameters;
    @XmlElement(name="Metadata",namespace=NS)
    private Elements metadata;
    // attributes and elements are not private for performance reasons
    // (JAXB can bypass reflection)
    @XmlAnyAttribute
    Map<QName,String> attributes;
    @XmlAnyElement
    List<Element> elements;


    @XmlType(name="address", namespace=W3CEndpointReference.NS)
    private static class Address {
        protected Address() {}
        @XmlValue
        String uri;
        @XmlAnyAttribute
        Map<QName,String> attributes;
    }


    @XmlType(name="elements", namespace=W3CEndpointReference.NS)
    private static class Elements {
        protected Elements() {}
        @XmlAnyElement
        List<Element> elements;
        @XmlAnyAttribute
        Map<QName,String> attributes;
    }

}
```

34.2 W3CEndpointReferenceBuilder.java

Secuencia 317: javax/xml/ws/wsaddressing/W3CEndpointReferenceBuilder.java

```
/*
 * Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.ws.wsaddressing;


import org.w3c.dom.Element;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.HashMap;
import javax.xml.namespace.QName;
import javax.xml.ws.WebServiceException;
import javax.xml.ws.spi.Provider;


/**
 * This class is used to build <code>W3CEndpointReference</code>
 * instances. The intended use of this clsss is for
 * an application component, for example a factory component,
 * to create an <code>W3CEndpointReference</code> for a
 * web service endpoint published by the same
 * Java EE application. It can also be used to create
 * <code>W3CEndpointReferences</code> for an Java SE based
 * endpoint by providing the <code>address</code> property.
 * <p>
 * When creating a <code>W3CEndpointReference</code> for an
 * endpoint that is not published by the same Java EE application,
 * the <code>address</code> property MUST be specified.
 * <p>
 * When creating a <code>W3CEndpointReference</code> for an endpoint
 * published by the same Java EE application, the <code>address</code>
 * property MAY be <code>null</code> but then the <code>serviceName</code>
```

```
 * and <code>endpointName</code> MUST specify an endpoint published by
 * the same Java EE application.
 * <p>
 * When the <code>wsdlDocumentLocation</code> is specified it MUST refer
 * to a valid WSDL document and the <code>serviceName</code> and
 * <code>endpointName</code> (if specified) MUST match a service and port
 * in the WSDL document.
 *
 * @since JAX-WS 2.1
 */
public final class W3CEndpointReferenceBuilder {
    /**
     * Creates a new <code>W3CEndpointReferenceBuilder</code> instance.
     */
    public W3CEndpointReferenceBuilder() {
        referenceParameters = new ArrayList<Element>();
        metadata = new ArrayList<Element>();
        attributes = new HashMap<QName, String>();
        elements = new ArrayList<Element>();
    }

    /**
     * Sets the <code>address</code> to the
     * <code>W3CEndpointReference</code> instance's
     * <code>wsa:Address</code>.
     * <p>
     * The <code>address</code> MUST be set to a non-<code>null</code>
     * value when building a <code>W3CEndpointReference</code> for a
     * web service endpoint that is not published by the same
     * Java EE application or when running on Java SE.
     *
     * @param address The address of the endpoint to be targeted
     *      by the returned <code>W3CEndpointReference</code>.
     *
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>address</code> set to the <code>wsa:Address</code>.
     */
    public W3CEndpointReferenceBuilder address(String address) {
        this.address = address;
        return this;
    }

    /**
     * Sets the <code>interfaceName</code> as the
     * <code>wsam:InterfaceName</code> element in the
     * <code>wsa:Metadata</code> element.
     *
     * See <a href="http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/#refmetadatfromepr">
     * 2.1 Referencing WSDL Metadata from an EPR</a> for more details.
     *
     * @param interfaceName The port type name of the endpoint to be targeted
     *      by the returned <code>W3CEndpointReference</code>.
     *
```

```
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>interfaceName</code> as <code>wsam:InterfaceName</code>
     *   element added to the <code>wsa:Metadata</code> element
     */
    public W3CEndpointReferenceBuilder interfaceName(QName interfaceName) {
        this.interfaceName = interfaceName;
        return this;
    }

    /**
     * Sets the <code>serviceName</code> as the
     * <code>wsam:ServiceName</code> element in the
     * <code>wsa:Metadata</code> element.
     *
     * See <a href="http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/#refmetadatfromepr">
     * 2.1 Referencing WSDL Metadata from an EPR</a> for more details.
     *
     * @param serviceName The service name of the endpoint to be targeted
     *      by the returned <code>W3CEndpointReference</code>.  This property
     *      may also be used with the <code>endpointName</code> (portName)
     *      property to lookup the <code>address</code> of a web service
     *      endpoint that is published by the same Java EE application.
     *
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>serviceName</code> as <code>wsam:ServiceName</code>
     *   element added to the <code>wsa:Metadata</code> element
     *
     */
    public W3CEndpointReferenceBuilder serviceName(QName serviceName) {
        this.serviceName = serviceName;
        return this;
    }

    /**
     * Sets the <code>endpointName</code> as
     * <code>wsam:ServiceName/@EndpointName</code> in the
     * <code>wsa:Metadata</code> element. This method can only be called
     * after the {@link #serviceName} method has been called.
     * <p>
     * See <a href="http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/#refmetadatfromepr">
     * 2.1 Referencing WSDL Metadata from an EPR</a> for more details.
     *
     * @param endpointName The name of the endpoint to be targeted
     *      by the returned <code>W3CEndpointReference</code>. The
     *      <code>endpointName</code> (portName) property may also be
     *      used with the <code>serviceName</code> property to lookup
     *      the <code>address</code> of a web service
     *      endpoint published by the same Java EE application.
     *
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>endpointName</code> as
     * <code>wsam:ServiceName/@EndpointName</code> in the
     * <code>wsa:Metadata</code> element.
```

```
     *
     * @throws IllegalStateException, if the <code>serviceName</code>
     * has not been set.
     * @throws IllegalArgumentException, if the <code>endpointName</code>'s
     * Namespace URI doesn't match <code>serviceName</code>'s Namespace URI
     *
     */
    public W3CEndpointReferenceBuilder endpointName(QName endpointName) {
        if (serviceName == null) {
            throw new IllegalStateException("The W3CEndpointReferenceBuilder's serviceName must be
                set before setting the endpointName: "+endpointName);
        }

        this.endpointName = endpointName;
        return this;
    }

    /**
     * Sets the <code>wsdlDocumentLocation</code> that will be referenced
     * as <code>wsa:Metadata/@wsdli:wsdlLocation</code>. The namespace name
     * for the wsdli:wsdlLocation's value can be taken from the WSDL itself.
     *
     * <p>
     * See <a href="http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/#refmetadatfromepr">
     * 2.1 Referencing WSDL Metadata from an EPR</a> for more details.
     *
     * @param wsdlDocumentLocation The location of the WSDL document to
     *      be referenced in the <code>wsa:Metadata</code> of the
     *     <code>W3CEndpointReference</code>.
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>wsdlDocumentLocation</code> that is to be referenced.
     */
    public W3CEndpointReferenceBuilder wsdlDocumentLocation(String wsdlDocumentLocation) {
        this.wsdlDocumentLocation = wsdlDocumentLocation;
        return this;
    }

    /**
     * Adds the <code>referenceParameter</code> to the
     * <code>W3CEndpointReference</code> instance
     * <code>wsa:ReferenceParameters</code> element.
     *
     * @param referenceParameter The element to be added to the
     *      <code>wsa:ReferenceParameters</code> element.
     *
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>referenceParameter</code> added to the
     *   <code>wsa:ReferenceParameters</code> element.
     *
     * @throws java.lang.IllegalArgumentException if <code>referenceParameter</code>
     * is <code>null</code>.
     */
    public W3CEndpointReferenceBuilder referenceParameter(Element referenceParameter) {
```

```
        if (referenceParameter == null)
            throw new java.lang.IllegalArgumentException("The referenceParameter cannot be null.");
        referenceParameters.add(referenceParameter);
        return this;
    }

    /**
     * Adds the <code>metadataElement</code> to the
     * <code>W3CEndpointReference</code> instance's
     * <code>wsa:Metadata</code> element.
     *
     * @param metadataElement The element to be added to the
     *      <code>wsa:Metadata</code> element.
     *
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the <code>metadataElement</code> added to the
     *    <code>wsa:Metadata</code> element.
     *
     * @throws java.lang.IllegalArgumentException if <code>metadataElement</code>
     * is <code>null</code>.
     */
    public W3CEndpointReferenceBuilder metadata(Element metadataElement) {
        if (metadataElement == null)
            throw new java.lang.IllegalArgumentException("The metadataElement cannot be null.");
        metadata.add(metadataElement);
        return this;
    }

    /**
     * Adds an extension element to the
     * <code>W3CEndpointReference</code> instance's
     * <code>wsa:EndpointReference</code> element.
     *
     * @param element The extension element to be added to the
     *   <code>W3CEndpointReference</code>
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the extension <code>element</code> added to the
     *    <code>W3CEndpointReference</code> instance.
     * @throws java.lang.IllegalArgumentException if <code>element</code>
     * is <code>null</code>.
     *
     * @since JAX-WS 2.2
     */
    public W3CEndpointReferenceBuilder element(Element element) {
        if (element == null) {
            throw new IllegalArgumentException("The extension element cannot be null.");
        }
        elements.add(element);
        return this;
    }

    /**
     * Adds an extension attribute to the
```

```
     * <code>W3CEndpointReference</code> instance's
     * <code>wsa:EndpointReference</code> element.
     *
     * @param name The name of the extension attribute to be added to the
     *   <code>W3CEndpointReference</code>
     * @param value extension attribute value
     * @return A <code>W3CEndpointReferenceBuilder</code> instance with
     *   the extension attribute added to the <code>W3CEndpointReference</code>
     *   instance.
     * @throws java.lang.IllegalArgumentException if <code>name</code>
     *   or <code>value</code> is <code>null</code>.
     *
     * @since JAX-WS 2.2
     */
    public W3CEndpointReferenceBuilder attribute(QName name, String value) {
        if (name == null || value == null) {
            throw new IllegalArgumentException("The extension attribute name or value cannot be
                null.");
        }
        attributes.put(name, value);
        return this;
    }

    /**
     * Builds a <code>W3CEndpointReference</code> from the accumulated
     * properties set on this <code>W3CEndpointReferenceBuilder</code>
     * instance.
     * <p>
     * This method can be used to create a <code>W3CEndpointReference</code>
     * for any endpoint by specifying the <code>address</code> property along
     * with any other desired properties.  This method
     * can also be used to create a <code>W3CEndpointReference</code> for
     * an endpoint that is published by the same Java EE application.
     * This method can automatically determine the <code>address</code> of
     * an endpoint published by the same Java EE application that is identified by the
     * <code>serviceName</code> and
     * <code>endpointName</code> properties.  If the <code>address</code> is
     * <code>null</code> and the <code>serviceName</code> and
     * <code>endpointName</code>
     * do not identify an endpoint published by the same Java EE application, a
     * <code>java.lang.IllegalStateException</code> MUST be thrown.
     *
     *
     * @return <code>W3CEndpointReference</code> from the accumulated
     * properties set on this <code>W3CEndpointReferenceBuilder</code>
     * instance. This method never returns <code>null</code>.
     *
     * @throws IllegalStateException
     *     <ul>
     *        <li>If the <code>address</code>, <code>serviceName</code> and
     *            <code>endpointName</code> are all <code>null</code>.
     *        <li>If the <code>serviceName</code> service is <code>null</code> and the
     *            <code>endpointName</code> is NOT <code>null</code>.
```

```
     *        <li>If the <code>address</code> property is <code>null</code> and
     *            the <code>serviceName</code> and <code>endpointName</code> do not
     *            specify a valid endpoint published by the same Java EE
     *            application.
     *        <li>If the <code>serviceName</code> is NOT <code>null</code>
     *             and is not present in the specified WSDL.
     *        <li>If the <code>endpointName</code> port is not <code>null</code> and it
     *             is not present in <code>serviceName</code> service in the WSDL.
     *        <li>If the <code>wsdlDocumentLocation</code> is NOT <code>null</code>
     *            and does not represent a valid WSDL.
     *     </ul>
     * @throws WebServiceException If an error occurs while creating the
     *                             <code>W3CEndpointReference</code>.
     *
     */
    public W3CEndpointReference build() {
        if (elements.isEmpty() && attributes.isEmpty() && interfaceName == null) {
            // 2.1 API
            return Provider.provider().createW3CEndpointReference(address,
                serviceName, endpointName, metadata, wsdlDocumentLocation,
                referenceParameters);
        }
        return Provider.provider().createW3CEndpointReference(address,
                interfaceName, serviceName, endpointName, metadata, wsdlDocumentLocation,
                referenceParameters, elements, attributes);
    }

    private String address;
    private List<Element> referenceParameters;
    private List<Element> metadata;
    private QName interfaceName;
    private QName serviceName;
    private QName endpointName;
    private String wsdlDocumentLocation;
    private Map<QName,String> attributes;
    private List<Element> elements;
}
```

34.3 package-info.java

Secuencia 318: javax/xml/ws/wsaddressing/package-info.java

```
/*
 * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

@javax.xml.bind.annotation.XmlSchema(namespace=W3CEndpointReference.NS,
                                     location="http://www.w3.org/2006/03/addressing/ws-addr.xsd")
package javax.xml.ws.wsaddressing;
```

35 Xml Xpath (javax.xml.xpath package)

35.1 SecuritySupport.java

Secuencia 319: javax/xml/xpath/SecuritySupport.java

```
/*
 * Copyright (c) 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import java.net.URL;
import java.security.*;
import java.net.*;
import java.io.*;
```

```java
import java.util.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


    ClassLoader getContextClassLoader() {
        return (ClassLoader)
                AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                try {
                    cl = Thread.currentThread().getContextClassLoader();
                } catch (SecurityException ex) { }
                return cl;
            }
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getURLInputStream(final URL url)
        throws IOException
    {
        try {
```

```
            return (InputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws IOException {
                        return url.openStream();
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (IOException)e.getException();
        }
    }

    URL getResourceAsURL(final ClassLoader cl,
                                        final String name)
    {
        return (URL)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    URL url;
                    if (cl == null) {
                        url = Object.class.getResource(name);
                    } else {
                        url = cl.getResource(name);
                    }
                    return url;
                }
            });
    }

    Enumeration getResources(final ClassLoader cl,
                                        final String name) throws IOException
    {
        try{
        return (Enumeration)
            AccessController.doPrivileged(new PrivilegedExceptionAction() {
                public Object run() throws IOException{
                    Enumeration enumeration;
                    if (cl == null) {
                        enumeration = ClassLoader.getSystemResources(name);
                    } else {
                        enumeration = cl.getResources(name);
                    }
                    return enumeration;
                }
            });
        }catch(PrivilegedActionException e){
            throw (IOException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                        final String name)
    {
        return (InputStream)
```

```
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

}
```

35.2 XPath.java

Secuencia 320: javax/xml/xpath/XPath.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;
```

```

import org.xml.sax.InputSource;
import javax.xml.namespace.QName;
import javax.xml.namespace.NamespaceContext;

/**
 * <p><code>XPath</code> provides access to the XPath evaluation environment and expressions.</p>
 *
 * <a name="XPath-evaluation"/>
 * <table border="1" cellpadding="2">
 *   <thead>
 *     <tr>
 *       <th colspan="2">Evaluation of XPath Expressions.</th>
 *     </tr>
 *   </thead>
 *   <tbody>
 *     <tr>
 *       <td>context</td>
 *       <td>
 *         If a request is made to evaluate the expression in the absence
 * of a context item, an empty document node will be used for the context.
 * For the purposes of evaluating XPath expressions, a DocumentFragment
 * is treated like a Document node.
 *      </td>
 *    </tr>
 *    <tr>
 *      <td>variables</td>
 *      <td>
 *        If the expression contains a variable reference, its value will be found through the {@link XPathVariableResolver}
 *        set with {@link #setXPathVariableResolver(XPathVariableResolver resolver)}.
 *        An {@link XPathExpressionException} is raised if the variable resolver is undefined or
 *        the resolver returns <code>null</code> for the variable.
 *        The value of a variable must be immutable through the course of any single evaluation.</p>
 *      </td>
 *    </tr>
 *    <tr>
 *      <td>functions</td>
 *      <td>
 *        If the expression contains a function reference, the function will be found through the {@link XPathFunctionResolver}
 *        set with {@link #setXPathFunctionResolver(XPathFunctionResolver resolver)}.
 *        An {@link XPathExpressionException} is raised if the function resolver is undefined or
 *        the function resolver returns <code>null</code> for the function.</p>
 *      </td>
 *    </tr>
 *    <tr>
 *      <td>QNames</td>
 *      <td>
 *        QNames in the expression are resolved against the XPath namespace context
 *        set with {@link #setNamespaceContext(NamespaceContext nsContext)}.
 *      </td>
```

```
 *    </tr>
 *    <tr>
 *      <td>result</td>
 *      <td>
 *        This result of evaluating an expression is converted to an instance of the desired return type.
 *        Valid return types are defined in {@link XPathConstants}.
 *        Conversion to the return type follows XPath conversion rules.</p>
 *      </td>
 *    </tr>
 * </table>
 *
 * <p>An XPath object is not thread-safe and not reentrant.
 * In other words, it is the application's responsibility to make
 * sure that one {@link XPath} object is not used from
 * more than one thread at any given time, and while the <code>evaluate</code>
 * method is invoked, applications may not recursively call
 * the <code>evaluate</code> method.
 * <p>
 *
 * @author  <a href="Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @see <a href="http://www.w3.org/TR/xpath">XML Path Language (XPath) Version 1.0</a>
 * @since 1.5
 */
public interface XPath {

        /**
         * <p>Reset this <code>XPath</code> to its original configuration.</p>
         *
         * <p><code>XPath</code> is reset to the same state as when it was created with
         * {@link XPathFactory#newXPath()}.
         * <code>reset()</code> is designed to allow the reuse of existing <code>XPath</code>s
         * thus saving resources associated with the creation of new <code>XPath</code>s.</p>
         *
         * <p>The reset <code>XPath</code> is not guaranteed to have the same {@link XPathFunctionResolver}, {@link XPathVariableResolver}
         * or {@link NamespaceContext} <code>Object</code>s, e.g. {@link Object#equals(Object obj)}.
         * It is guaranteed to have a functionally equal <code>XPathFunctionResolver</code>, <code>XPathVariableResolver</code>
         * and <code>NamespaceContext</code>.</p>
         */
        public void reset();

    /**
     * <p>Establish a variable resolver.</p>
     *
     * <p>A <code>NullPointerException</code> is thrown if <code>resolver</code> is <code>null</code>.</p>
     *
     * @param resolver Variable resolver.
     *
```

```
     *  @throws NullPointerException If <code>resolver</code> is <code>null</code>.
     */
    public void setXPathVariableResolver(XPathVariableResolver resolver);

    /**
       * <p>Return the current variable resolver.</p>
       *
       * <p><code>null</code> is returned in no variable resolver is in effect.</p>
       *
       * @return Current variable resolver.
       */
    public XPathVariableResolver getXPathVariableResolver();

    /**
       * <p>Establish a function resolver.</p>
       *
       * <p>A <code>NullPointerException</code> is thrown if <code>resolver</code> is <code>null</
           code>.</p>
       *
       * @param resolver XPath function resolver.
       *
       * @throws NullPointerException If <code>resolver</code> is <code>null</code>.
       */
    public void setXPathFunctionResolver(XPathFunctionResolver resolver);

    /**
       * <p>Return the current function resolver.</p>
       *
       * <p><code>null</code> is returned in no function resolver is in effect.</p>
       *
       * @return Current function resolver.
       */
    public XPathFunctionResolver getXPathFunctionResolver();

    /**
       * <p>Establish a namespace context.</p>
       *
       * <p>A <code>NullPointerException</code> is thrown if <code>nsContext</code> is <code>null</
           code>.</p>
       *
       * @param nsContext Namespace context to use.
       *
       * @throws NullPointerException If <code>nsContext</code> is <code>null</code>.
       */
    public void setNamespaceContext(NamespaceContext nsContext);

    /**
       * <p>Return the current namespace context.</p>
       *
       * <p><code>null</code> is returned in no namespace context is in effect.</p>
       *
       * @return Current Namespace context.
       */
```

```
    public NamespaceContext getNamespaceContext();

    /**
       * <p>Compile an XPath expression for later evaluation.</p>
       *
       * <p>If <code>expression</code> contains any {@link XPathFunction}s,
       * they must be available via the {@link XPathFunctionResolver}.
       * An {@link XPathExpressionException} will be thrown if the
       * <code>XPathFunction</code>
       * cannot be resovled with the <code>XPathFunctionResolver</code>.</p>
       *
       * <p>If <code>expression</code> contains any variables, the
       * {@link XPathVariableResolver} in effect
       * <strong>at compile time</strong> will be used to resolve them.</p>
       *
       * <p>If <code>expression</code> is <code>null</code>, a <code>NullPointerException</code> is
           thrown.</p>
       *
       * @param expression The XPath expression.
       *
       * @return Compiled XPath expression.

       * @throws XPathExpressionException If <code>expression</code> cannot be compiled.
       * @throws NullPointerException If <code>expression</code> is <code>null</code>.
       */
    public XPathExpression compile(String expression)
        throws XPathExpressionException;

    /**
     * <p>Evaluate an <code>XPath</code> expression in the specified context and return the result
         as the specified type.</p>
     *
     * <p>See <a href="#XPath-evaluation">Evaluation of XPath Expressions</a> for context item
         evaluation,
     * variable, function and <code>QName</code> resolution and return type conversion.</p>
     *
     * <p>If <code>returnType</code> is not one of the types defined in {@link XPathConstants} (
     * {@link XPathConstants#NUMBER NUMBER},
     * {@link XPathConstants#STRING STRING},
     * {@link XPathConstants#BOOLEAN BOOLEAN},
     * {@link XPathConstants#NODE NODE} or
     * {@link XPathConstants#NODESET NODESET})
     * then an <code>IllegalArgumentException</code> is thrown.</p>
     *
     * <p>If a <code>null</code> value is provided for
     * <code>item</code>, an empty document will be used for the
     * context.
     * If <code>expression</code> or <code>returnType</code> is <code>null</code>, then a
     * <code>NullPointerException</code> is thrown.</p>
     *
     * @param expression The XPath expression.
     * @param item The starting context (a node, for example).
     * @param returnType The desired return type.
```

```
     *
     * @return Result of evaluating an XPath expression as an <code>Object</code> of <code>
         returnType</code>.
     *
     * @throws XPathExpressionException If <code>expression</code> cannot be evaluated.
     * @throws IllegalArgumentException If <code>returnType</code> is not one of the types defined
         in {@link XPathConstants}.
     * @throws NullPointerException If <code>expression</code> or <code>returnType</code> is <code>
         null</code>.
     */
    public Object evaluate(String expression, Object item, QName returnType)
        throws XPathExpressionException;

    /**
     * <p>Evaluate an XPath expression in the specified context and return the result as a <code>
         String</code>.</p>
     *
     * <p>This method calls {@link #evaluate(String expression, Object item, QName returnType)}
         with a <code>returnType</code> of
     * {@link XPathConstants#STRING}.</p>
     *
     * <p>See <a href="#XPath-evaluation">Evaluation of XPath Expressions</a> for context item
         evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If a <code>null</code> value is provided for
     * <code>item</code>, an empty document will be used for the
     * context.
     * If <code>expression</code> is <code>null</code>, then a <code>NullPointerException</code> is
          thrown.</p>
     *
     * @param expression The XPath expression.
     * @param item The starting context (a node, for example).
     *
     * @return The <code>String</code> that is the result of evaluating the expression and
     *   converting the result to a <code>String</code>.
     *
     * @throws XPathExpressionException If <code>expression</code> cannot be evaluated.
     * @throws NullPointerException If <code>expression</code> is <code>null</code>.
     */
    public String evaluate(String expression, Object item)
        throws XPathExpressionException;

    /**
     * <p>Evaluate an XPath expression in the context of the specified <code>InputSource</code>
     * and return the result as the specified type.</p>
     *
     * <p>This method builds a data model for the {@link InputSource} and calls
     * {@link #evaluate(String expression, Object item, QName returnType)} on the resulting
         document object.</p>
     *
     * <p>See <a href="#XPath-evaluation">Evaluation of XPath Expressions</a> for context item
         evaluation,
```

```
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If <code>returnType</code> is not one of the types defined in {@link XPathConstants},
     * then an <code>IllegalArgumentException</code> is thrown.</p>
     *
     * <p>If <code>expression</code>, <code>source</code> or <code>returnType</code> is <code>null
       </code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param expression The XPath expression.
     * @param source The input source of the document to evaluate over.
     * @param returnType The desired return type.
     *
     * @return The <code>Object</code> that encapsulates the result of evaluating the expression.
     *
     * @throws XPathExpressionException If expression cannot be evaluated.
     * @throws IllegalArgumentException If <code>returnType</code> is not one of the types defined
       in {@link XPathConstants}.
     * @throws NullPointerException If <code>expression</code>, <code>source</code> or <code>
       returnType</code>
     *   is <code>null</code>.
     */
    public Object evaluate(
        String expression,
        InputSource source,
        QName returnType)
        throws XPathExpressionException;

    /**
     * <p>Evaluate an XPath expression in the context of the specified <code>InputSource</code>
     * and return the result as a <code>String</code>.</p>
     *
     * <p>This method calls {@link #evaluate(String expression, InputSource source, QName
       returnType)} with a
     * <code>returnType</code> of {@link XPathConstants#STRING}.</p>
     *
     * <p>See <a href="#XPath-evaluation">Evaluation of XPath Expressions</a> for context item
       evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If <code>expression</code> or <code>source</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param expression The XPath expression.
     * @param source The <code>InputSource</code> of the document to evaluate over.
     *
     * @return The <code>String</code> that is the result of evaluating the expression and
     *   converting the result to a <code>String</code>.
     *
     * @throws XPathExpressionException If expression cannot be evaluated.
     * @throws NullPointerException If <code>expression</code> or <code>source</code> is <code>null
       </code>.
     */
```

```
    public String evaluate(String expression, InputSource source)
        throws XPathExpressionException;
}
```

35.3 XPathConstants.java

Secuencia 321: javax/xml/xpath/XPathConstants.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import javax.xml.namespace.QName;

/**
 * <p>XPath constants.</p>
 *
 * @author <a href="mailto:Norman.Walsh@Sun.COM">Norman Walsh</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.COM">Jeff Suttor</a>
 * @see <a href="http://www.w3.org/TR/xpath">XML Path Language (XPath) Version 1.0</a>
 * @since 1.5
 */
public class XPathConstants {

    /**
     * <p>Private constructor to prevent instantiation.</p>
     */
    private XPathConstants() { }

    /**
     * <p>The XPath 1.0 number data type.</p>
```

```
     *
     * <p>Maps to Java {@link Double}.</p>
     */
    public static final QName NUMBER = new QName("http://www.w3.org/1999/XSL/Transform", "NUMBER");

    /**
     * <p>The XPath 1.0 string data type.</p>
     *
     * <p>Maps to Java {@link String}.</p>
     */
    public static final QName STRING = new QName("http://www.w3.org/1999/XSL/Transform", "STRING");

    /**
     * <p>The XPath 1.0 boolean data type.</p>
     *
     * <p>Maps to Java {@link Boolean}.</p>
     */
    public static final QName BOOLEAN = new QName("http://www.w3.org/1999/XSL/Transform", "BOOLEAN"
        );

    /**
     * <p>The XPath 1.0 NodeSet data type.</p>
     *
     * <p>Maps to Java {@link org.w3c.dom.NodeList}.</p>
     */
    public static final QName NODESET = new QName("http://www.w3.org/1999/XSL/Transform", "NODESET"
        );

    /**
     * <p>The XPath 1.0 NodeSet data type.
     *
     * <p>Maps to Java {@link org.w3c.dom.Node}.</p>
     */
    public static final QName NODE = new QName("http://www.w3.org/1999/XSL/Transform", "NODE");

    /**
     * <p>The URI for the DOM object model, "http://java.sun.com/jaxp/xpath/dom".</p>
     */
    public static final String DOM_OBJECT_MODEL = "http://java.sun.com/jaxp/xpath/dom";
}
```

35.4 XPathException.java

Secuencia 322: javax/xml/xpath/XPathException.java

```
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import java.io.PrintWriter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.ObjectStreamField;
import java.io.InvalidClassException;

/**
 * <code>XPathException</code> represents a generic XPath exception.</p>
 *
 * @author  <a href="Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author <a href="mailto:Jeff.Suttor@Sun.COM">Jeff Suttor</a>
 * @since 1.5
 */
public class XPathException extends Exception {

    private static final ObjectStreamField[] serialPersistentFields = {
        new ObjectStreamField( "cause", Throwable.class )
    };

    /**
     * <p>Stream Unique Identifier.</p>
     */
    private static final long serialVersionUID = -1837080260374986980L;

    /**
     * <p>Constructs a new <code>XPathException</code>
     * with the specified detail <code>message</code>.</p>
     *
     * <p>The <code>cause</code> is not initialized.</p>
     *
     * <p>If <code>message</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param message The detail message.
```

```
     *
     * @throws NullPointerException When <code>message</code> is
     *   <code>null</code>.
     */
    public XPathException(String message) {
        super(message);
        if ( message == null ) {
            throw new NullPointerException ( "message can't be null");
        }
    }

    /**
     * <p>Constructs a new <code>XPathException</code>
     * with the specified <code>cause</code>.</p>
     *
     * <p>If <code>cause</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param cause The cause.
     *
     * @throws NullPointerException if <code>cause</code> is <code>null</code>.
     */
    public XPathException(Throwable cause) {
        super(cause);
        if ( cause == null ) {
            throw new NullPointerException ( "cause can't be null");
        }
    }

    /**
     * <p>Get the cause of this XPathException.</p>
     *
     * @return Cause of this XPathException.
     */
    public Throwable getCause() {
        return super.getCause();
    }

    /**
     * Writes "cause" field to the stream.
     * The cause is got from the parent class.
     *
     * @param out stream used for serialization.
     * @throws IOException thrown by <code>ObjectOutputStream</code>
     *
     */
    private void writeObject(ObjectOutputStream out)
            throws IOException
    {
        ObjectOutputStream.PutField fields = out.putFields();
        fields.put("cause", (Throwable) super.getCause());
        out.writeFields();
    }
```

```java

    /**
     * Reads the "cause" field from the stream.
     * And initializes the "cause" if it wasn't
     * done before.
     *
     * @param in stream used for deserialization
     * @throws IOException thrown by <code>ObjectInputStream</code>
     * @throws ClassNotFoundException  thrown by <code>ObjectInputStream</code>
     */
    private void readObject(ObjectInputStream in)
            throws IOException, ClassNotFoundException
    {
        ObjectInputStream.GetField fields = in.readFields();
        Throwable scause = (Throwable) fields.get("cause", null);

        if (super.getCause() == null && scause != null) {
            try {
                super.initCause(scause);
            } catch(IllegalStateException e) {
                throw new InvalidClassException("Inconsistent state: two causes");
            }
        }
    }

    /**
     * <p>Print stack trace to specified <code>PrintStream</code>.</p>
     *
     * @param s Print stack trace to this <code>PrintStream</code>.
     */
    public void printStackTrace(java.io.PrintStream s) {
        if (getCause() != null) {
            getCause().printStackTrace(s);
          s.println("--------------- linked to ------------------");
        }

        super.printStackTrace(s);
    }

    /**
     * <p>Print stack trace to <code>System.err</code>.</p>
     */
    public void printStackTrace() {
        printStackTrace(System.err);
    }

    /**
     * <p>Print stack trace to specified <code>PrintWriter</code>.</p>
     *
     * @param s Print stack trace to this <code>PrintWriter</code>.
     */
    public void printStackTrace(PrintWriter s) {

```

```
        if (getCause() != null) {
            getCause().printStackTrace(s);
          s.println("--------------- linked to ------------------");
        }

        super.printStackTrace(s);
    }
}
```

35.5 XPathExpression.java

Secuencia 323: javax/xml/xpath/XPathExpression.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import org.xml.sax.InputSource;
import javax.xml.namespace.QName;

/**
 * <p><code>XPathExpression</code> provides access to compiled XPath expressions.</p>
 *
 * <a name="XPathExpression-evaluation"/>
 * <table border="1" cellpadding="2">
 *   <thead>
 *     <tr>
 *       <th colspan="2">Evaluation of XPath Expressions.</th>
 *     </tr>
 *   </thead>
 *   <tbody>
```

```
*     <tr>
*       <td>context</td>
*       <td>
*         If a request is made to evaluate the expression in the absence
* of a context item, an empty document node will be used for the context.
* For the purposes of evaluating XPath expressions, a DocumentFragment
* is treated like a Document node.
*      </td>
*    </tr>
*    <tr>
*      <td>variables</td>
*      <td>
*        If the expression contains a variable reference, its value will be found through the {@link XPathVariableResolver}.
*        An {@link XPathExpressionException} is raised if the variable resolver is undefined or
*        the resolver returns <code>null</code> for the variable.
*        The value of a variable must be immutable through the course of any single evaluation.</p>
*      </td>
*    </tr>
*    <tr>
*      <td>functions</td>
*      <td>
*        If the expression contains a function reference, the function will be found through the {@link XPathFunctionResolver}.
*        An {@link XPathExpressionException} is raised if the function resolver is undefined or
*        the function resolver returns <code>null</code> for the function.</p>
*      </td>
*    </tr>
*    <tr>
*      <td>QNames</td>
*      <td>
*        QNames in the expression are resolved against the XPath namespace context.
*      </td>
*    </tr>
*    <tr>
*      <td>result</td>
*      <td>
*        This result of evaluating an expression is converted to an instance of the desired return type.
*        Valid return types are defined in {@link XPathConstants}.
*        Conversion to the return type follows XPath conversion rules.</p>
*      </td>
*    </tr>
* </table>
*
* <p>An XPath expression is not thread-safe and not reentrant.
* In other words, it is the application's responsibility to make
* sure that one {@link XPathExpression} object is not used from
* more than one thread at any given time, and while the <code>evaluate</code>
* method is invoked, applications may not recursively call
* the <code>evaluate</code> method.
* <p>
```

```
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @see <a href="http://www.w3.org/TR/xpath#section-Expressions">XML Path Language (XPath) Version 1.0, Expressions</a>
 * @since 1.5
 */
public interface XPathExpression {

    /**
     * <p>Evaluate the compiled XPath expression in the specified context and return the result as the specified type.</p>
     *
     * <p>See <a href="#XPathExpression-evaluation">Evaluation of XPath Expressions</a> for context item evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If <code>returnType</code> is not one of the types defined in {@link XPathConstants},
     * then an <code>IllegalArgumentException</code> is thrown.</p>
     *
     * <p>If a <code>null</code> value is provided for
     * <code>item</code>, an empty document will be used for the
     * context.
     * If <code>returnType</code> is <code>null</code>, then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param item The starting context (a node, for example).
     * @param returnType The desired return type.
     *
     * @return The <code>Object</code> that is the result of evaluating the expression and converting the result to
     *   <code>returnType</code>.
     *
     * @throws XPathExpressionException If the expression cannot be evaluated.
     * @throws IllegalArgumentException If <code>returnType</code> is not one of the types defined in {@link XPathConstants}.
     * @throws NullPointerException If  <code>returnType</code> is <code>null</code>.
     */
    public Object evaluate(Object item, QName returnType)
        throws XPathExpressionException;

    /**
     * <p>Evaluate the compiled XPath expression in the specified context and return the result as a <code>String</code>.</p>
     *
     * <p>This method calls {@link #evaluate(Object item, QName returnType)} with a <code>returnType</code> of
     * {@link XPathConstants#STRING}.</p>
     *
     * <p>See <a href="#XPathExpression-evaluation">Evaluation of XPath Expressions</a> for context item evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
```

```
     * <p>If a <code>null</code> value is provided for
     * <code>item</code>, an empty document will be used for the
     * context.
     *
     * @param item The starting context (a node, for example).
     *
     * @return The <code>String</code> that is the result of evaluating the expression and converting the result to a
     *   <code>String</code>.
     *
     * @throws XPathExpressionException If the expression cannot be evaluated.
     */
    public String evaluate(Object item)
        throws XPathExpressionException;

    /**
     * <p>Evaluate the compiled XPath expression in the context of the specified <code>InputSource</code> and return the result as the
     * specified type.</p>
     *
     * <p>This method builds a data model for the {@link InputSource} and calls
     * {@link #evaluate(Object item, QName returnType)} on the resulting document object.</p>
     *
     * <p>See <a href="#XPathExpression-evaluation">Evaluation of XPath Expressions</a> for context item evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If <code>returnType</code> is not one of the types defined in {@link XPathConstants},
     * then an <code>IllegalArgumentException</code> is thrown.</p>
     *
     * <p>If <code>source</code> or <code>returnType</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param source The <code>InputSource</code> of the document to evaluate over.
     * @param returnType The desired return type.
     *
     * @return The <code>Object</code> that is the result of evaluating the expression and converting the result to
     *   <code>returnType</code>.
     *
     * @throws XPathExpressionException If the expression cannot be evaluated.
     * @throws IllegalArgumentException If <code>returnType</code> is not one of the types defined in {@link XPathConstants}.
     * @throws NullPointerException If  <code>source</code> or <code>returnType</code> is <code>null</code>.
     */
    public Object evaluate(InputSource source, QName returnType)
        throws XPathExpressionException;

    /**
     * <p>Evaluate the compiled XPath expression in the context of the specified <code>InputSource</code> and return the result as a
     * <code>String</code>.</p>
```

```
     *
     * <p>This method calls {@link #evaluate(InputSource source, QName returnType)} with a <code>
         returnType</code> of
     * {@link XPathConstants#STRING}.</p>
     *
     * <p>See <a href="#XPathExpression-evaluation">Evaluation of XPath Expressions</a> for context
          item evaluation,
     * variable, function and QName resolution and return type conversion.</p>
     *
     * <p>If <code>source</code> is <code>null</code>, then a <code>NullPointerException</code> is
         thrown.</p>
     *
     * @param source The <code>InputSource</code> of the document to evaluate over.
     *
     * @return The <code>String</code> that is the result of evaluating the expression and
         converting the result to a
     *   <code>String</code>.
     *
     * @throws XPathExpressionException If the expression cannot be evaluated.
     * @throws NullPointerException If  <code>source</code> is <code>null</code>.
     */
    public String evaluate(InputSource source)
        throws XPathExpressionException;
}
```

35.6 XPathExpressionException.java

Secuencia 324: javax/xml/xpath/XPathExpressionException.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```java
package javax.xml.xpath;

/**
 * <code>XPathExpressionException</code> represents an error in an XPath expression.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */
public class XPathExpressionException extends XPathException {

    /**
     * <p>Stream Unique Identifier.</p>
     */
    private static final long serialVersionUID = -1837080260374986980L;

    /**
     * <p>Constructs a new <code>XPathExpressionException</code>
     * with the specified detail <code>message</code>.</p>
     *
     * <p>The <code>cause</code> is not initialized.</p>
     *
     * <p>If <code>message</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param message The detail message.
     *
     * @throws NullPointerException When <code>message</code> is
     *   <code>null</code>.
     */
    public XPathExpressionException(String message) {
        super(message);
    }

    /**
     * <p>Constructs a new <code>XPathExpressionException</code>
     * with the specified <code>cause</code>.</p>
     *
     * <p>If <code>cause</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param cause The cause.
     *
     * @throws NullPointerException if <code>cause</code> is <code>null</code>.
     */
    public XPathExpressionException(Throwable cause) {
        super(cause);
    }
}
```

35.7 XPathFactory.java

Secuencia 325: javax/xml/xpath/XPathFactory.java

```java
/*
 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

/**
 * <p>An <code>XPathFactory</code> instance can be used to create
 * {@link javax.xml.xpath.XPath} objects.</p>
 *
 *<p>See {@link #newInstance(String uri)} for lookup mechanism.</p>
 *
 * <p>The {@link XPathFactory} class is not thread-safe. In other words,
 * it is the application's responsibility to ensure that at most
 * one thread is using a {@link XPathFactory} object at any
 * given moment. Implementations are encouraged to mark methods
 * as <code>synchronized</code> to protect themselves from broken clients.
 *
 * <p>{@link XPathFactory} is not re-entrant. While one of the
 * <code>newInstance</code> methods is being invoked, applications
 * may not attempt to recursively invoke a <code>newInstance</code> method,
 * even from the same thread.
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 *
 * @since 1.5
 */
public abstract class XPathFactory {


    /**
```

```
     * <p>The default property name according to the JAXP spec.</p>
     */
    public static final String DEFAULT_PROPERTY_NAME = "javax.xml.xpath.XPathFactory";

    /**
     * <p>Default Object Model URI.</p>
     */
    public static final String DEFAULT_OBJECT_MODEL_URI = "http://java.sun.com/jaxp/xpath/dom";

    /**
     *<p> Take care of restrictions imposed by java security model </p>
     */
    private static SecuritySupport ss = new SecuritySupport() ;

    /**
     * <p>Protected constructor as {@link #newInstance()} or {@link #newInstance(String uri)}
     * or {@link #newInstance(String uri, String factoryClassName, ClassLoader classLoader)}
     * should be used to create a new instance of an <code>XPathFactory</code>.</p>
     */
    protected XPathFactory() {
    }

    /**
     * <p>Get a new <code>XPathFactory</code> instance using the default object model,
     * {@link #DEFAULT_OBJECT_MODEL_URI},
     * the W3C DOM.</p>
     *
     * <p>This method is functionally equivalent to:</p>
     * <pre>
     *   newInstance(DEFAULT_OBJECT_MODEL_URI)
     * </pre>
     *
     * <p>Since the implementation for the W3C DOM is always available, this method will never fail
         .</p>
     *
     * @return Instance of an <code>XPathFactory</code>.
     *
     * @throws RuntimeException When there is a failure in creating an
     *   <code>XPathFactory</code> for the default object model.
     */
    public static XPathFactory newInstance() {

        try {
                return newInstance(DEFAULT_OBJECT_MODEL_URI);
        } catch (XPathFactoryConfigurationException xpathFactoryConfigurationException) {
                throw new RuntimeException(
                        "XPathFactory#newInstance() failed to create an XPathFactory for the
                            default object model: "
                        + DEFAULT_OBJECT_MODEL_URI
                        + " with the XPathFactoryConfigurationException: "
                        + xpathFactoryConfigurationException.toString()
                );
        }
```

```
    }

    /**
    * <p>Get a new <code>XPathFactory</code> instance using the specified object model.</p>
    *
    * <p>To find a <code>XPathFactory</code> object,
    * this method looks the following places in the following order where "the class loader" refers to the context class loader:</p>
    * <ol>
    *   <li>
    *     If the system property {@link #DEFAULT_PROPERTY_NAME} + ":uri" is present,
    *     where uri is the parameter to this method, then its value is read as a class name.
    *     The method will try to create a new instance of this class by using the class loader,
    *     and returns it if it is successfully created.
    *   </li>
    *   <li>
    *     ${java.home}/lib/jaxp.properties is read and the value associated with the key being the system property above is looked for.
    *     If present, the value is processed just like above.
    *   </li>
    *   <li>
    *     Use the service-provider loading facilities, defined by the
    *     {@link java.util.ServiceLoader} class, to attempt to locate and load an
    *     implementation of the service using the {@linkplain
    *     java.util.ServiceLoader#load(java.lang.Class) default loading mechanism}:
    *     the service-provider loading facility will use the {@linkplain
    *     java.lang.Thread#getContextClassLoader() current thread's context class loader}
    *     to attempt to load the service. If the context class
    *     loader is null, the {@linkplain
    *     ClassLoader#getSystemClassLoader() system class loader} will be used.
    *     <br>
    *     Each potential service provider is required to implement the method
    *     {@link #isObjectModelSupported(String objectModel)}.
    *     The first service provider found that supports the specified object
    *     model is returned.
    *     <br>
    *     In case of {@link java.util.ServiceConfigurationError} an
    *     {@link XPathFactoryConfigurationException} will be thrown.
    *   </li>
    *   <li>
    *     Platform default <code>XPathFactory</code> is located in a platform specific way.
    *     There must be a platform default XPathFactory for the W3C DOM, i.e. {@link #DEFAULT_OBJECT_MODEL_URI}.
    *   </li>
    * </ol>
    * <p>If everything fails, an <code>XPathFactoryConfigurationException</code> will be thrown.</p>
    *
    * <p>Tip for Trouble-shooting:</p>
    * <p>See {@link java.util.Properties#load(java.io.InputStream)} for exactly how a property file is parsed.
    * In particular, colons ':' need to be escaped in a property file, so make sure the URIs are properly escaped in it.
```

```
    * For example:</p>
    * <pre>
    *   http\://java.sun.com/jaxp/xpath/dom=org.acme.DomXPathFactory
    * </pre>
    *
    * @param uri Identifies the underlying object model.
    *   The specification only defines the URI {@link #DEFAULT_OBJECT_MODEL_URI},
    *   <code>http://java.sun.com/jaxp/xpath/dom</code> for the W3C DOM,
    *   the org.w3c.dom package, and implementations are free to introduce other URIs for other
        object models.
    *
    * @return Instance of an <code>XPathFactory</code>.
    *
    * @throws XPathFactoryConfigurationException If the specified object model
    *      is unavailable, or if there is a configuration error.
    * @throws NullPointerException If <code>uri</code> is <code>null</code>.
    * @throws IllegalArgumentException If <code>uri</code> is <code>null</code>
    *   or <code>uri.length() == 0</code>.
    */
    public static XPathFactory newInstance(final String uri)
        throws XPathFactoryConfigurationException {

        if (uri == null) {
            throw new NullPointerException(
                    "XPathFactory#newInstance(String uri) cannot be called with uri == null");
        }

        if (uri.length() == 0) {
            throw new IllegalArgumentException(
                    "XPathFactory#newInstance(String uri) cannot be called with uri == \"\"");
        }

        ClassLoader classLoader = ss.getContextClassLoader();

        if (classLoader == null) {
            //use the current class loader
            classLoader = XPathFactory.class.getClassLoader();
        }

        XPathFactory xpathFactory = new XPathFactoryFinder(classLoader).newFactory(uri);

        if (xpathFactory == null) {
            throw new XPathFactoryConfigurationException(
                    "No XPathFactory implementation found for the object model: "
                    + uri);
        }

        return xpathFactory;
    }

    /**
     * <p>Obtain a new instance of a <code>XPathFactory</code> from a factory class name. <code>
         XPathFactory</code>
```

```
     * is returned if specified factory class supports the specified object model.
     * This function is useful when there are multiple providers in the classpath.
     * It gives more control to the application as it can specify which provider
     * should be loaded.</p>
     *
     *
     * <h2>Tip for Trouble-shooting</h2>
     * <p>Setting the <code>jaxp.debug</code> system property will cause
     * this method to print a lot of debug messages
     * to <code>System.err</code> about what it is doing and where it is looking at.</p>
     *
     * <p> If you have problems try:</p>
     * <pre>
     * java -Djaxp.debug=1 YourProgram ....
     * </pre>
     *
     * @param uri         Identifies the underlying object model. The specification only defines the URI
     *                    {@link #DEFAULT_OBJECT_MODEL_URI},<code>http://java.sun.com/jaxp/xpath/dom</code>
     *                    for the W3C DOM, the org.w3c.dom package, and implementations are free to introduce
     *                    other URIs for other object models.
     *
     * @param factoryClassName fully qualified factory class name that provides implementation of <code>javax.xml.xpath.XPathFactory</code>.
     *
     * @param classLoader <code>ClassLoader</code> used to load the factory class. If <code>null</code>
     *                     current <code>Thread</code>'s context classLoader is used to load the factory class.
     *
     *
     * @return New instance of a <code>XPathFactory</code>
     *
     * @throws XPathFactoryConfigurationException
     *                   if <code>factoryClassName</code> is <code>null</code>, or
     *                   the factory class cannot be loaded, instantiated
     *                   or the factory class does not support the object model specified
     *                   in the <code>uri</code> parameter.
     *
     * @throws NullPointerException If <code>uri</code> is <code>null</code>.
     * @throws IllegalArgumentException If <code>uri</code> is <code>null</code>
     *          or <code>uri.length() == 0</code>.
     *
     * @see #newInstance()
     * @see #newInstance(String uri)
     *
     * @since 1.6
     */
    public static XPathFactory newInstance(String uri, String factoryClassName, ClassLoader classLoader)
        throws XPathFactoryConfigurationException{
```

```
        ClassLoader cl = classLoader;

        if (uri == null) {
            throw new NullPointerException(
                    "XPathFactory#newInstance(String uri) cannot be called with uri == null");
        }

        if (uri.length() == 0) {
            throw new IllegalArgumentException(
                    "XPathFactory#newInstance(String uri) cannot be called with uri == \"\"");
        }

        if (cl == null) {
            cl = ss.getContextClassLoader();
        }

        XPathFactory f = new XPathFactoryFinder(cl).createInstance(factoryClassName);

        if (f == null) {
            throw new XPathFactoryConfigurationException(
                    "No XPathFactory implementation found for the object model: "
                    + uri);
        }
        //if this factory supports the given schemalanguage return this factory else thrown
            exception
        if (f.isObjectModelSupported(uri)) {
            return f;
        } else {
            throw new XPathFactoryConfigurationException("Factory "
                    + factoryClassName + " doesn't support given " + uri
                    + " object model");
        }

    }

    /**
     * <p>Is specified object model supported by this <code>XPathFactory</code>?</p>
     *
     * @param objectModel Specifies the object model which the returned <code>XPathFactory</code>
         will understand.
     *
     * @return <code>true</code> if <code>XPathFactory</code> supports <code>objectModel</code>,
         else <code>false</code>.
     *
     * @throws NullPointerException If <code>objectModel</code> is <code>null</code>.
     * @throws IllegalArgumentException If <code>objectModel.length() == 0</code>.
     */
    public abstract boolean isObjectModelSupported(String objectModel);

    /**
     * <p>Set a feature for this <code>XPathFactory</code> and
     * <code>XPath</code>s created by this factory.</p>
     *
```

```
     * <p>
     * Feature names are fully qualified {@link java.net.URI}s.
     * Implementations may define their own features.
     * An {@link XPathFactoryConfigurationException} is thrown if this
     * <code>XPathFactory</code> or the <code>XPath</code>s
     * it creates cannot support the feature.
     * It is possible for an <code>XPathFactory</code> to expose a feature value
     * but be unable to change its state.
     * </p>
     *
     * <p>
     * All implementations are required to support the {@link javax.xml.XMLConstants#
         FEATURE_SECURE_PROCESSING} feature.
     * When the feature is <code>true</code>, any reference to  an external function is an error.
     * Under these conditions, the implementation must not call the {@link XPathFunctionResolver}
     * and must throw an {@link XPathFunctionException}.
     * </p>
     *
     * @param name Feature name.
     * @param value Is feature state <code>true</code> or <code>false</code>.
     *
     * @throws XPathFactoryConfigurationException if this <code>XPathFactory</code> or the <code>
         XPath</code>s
     *   it creates cannot support this feature.
     * @throws NullPointerException if <code>name</code> is <code>null</code>.
     */
    public abstract void setFeature(String name, boolean value)
            throws XPathFactoryConfigurationException;

    /**
     * <p>Get the state of the named feature.</p>
     *
     * <p>
     * Feature names are fully qualified {@link java.net.URI}s.
     * Implementations may define their own features.
     * An {@link XPathFactoryConfigurationException} is thrown if this
     * <code>XPathFactory</code> or the <code>XPath</code>s
     * it creates cannot support the feature.
     * It is possible for an <code>XPathFactory</code> to expose a feature value
     * but be unable to change its state.
     * </p>
     *
     * @param name Feature name.
     *
     * @return State of the named feature.
     *
     * @throws XPathFactoryConfigurationException if this
     *   <code>XPathFactory</code> or the <code>XPath</code>s
     *   it creates cannot support this feature.
     * @throws NullPointerException if <code>name</code> is <code>null</code>.
     */
    public abstract boolean getFeature(String name)
            throws XPathFactoryConfigurationException;
```

```

    /**
     * <p>Establish a default variable resolver.</p>
     *
     * <p>Any <code>XPath</code> objects constructed from this factory will use
     * the specified resolver by default.</p>
     *
     * <p>A <code>NullPointerException</code> is thrown if <code>resolver</code>
     * is <code>null</code>.</p>
     *
     * @param resolver Variable resolver.
     *
     * @throws NullPointerException If <code>resolver</code> is
     *   <code>null</code>.
     */
    public abstract void setXPathVariableResolver(XPathVariableResolver resolver);

    /**
     * <p>Establish a default function resolver.</p>
     *
     * <p>Any <code>XPath</code> objects constructed from this factory will
     * use the specified resolver by default.</p>
     *
     * <p>A <code>NullPointerException</code> is thrown if
     * <code>resolver</code> is <code>null</code>.</p>
     *
     * @param resolver XPath function resolver.
     *
     * @throws NullPointerException If <code>resolver</code> is
     *   <code>null</code>.
     */
    public abstract void setXPathFunctionResolver(XPathFunctionResolver resolver);

    /**
    * <p>Return a new <code>XPath</code> using the underlying object
    * model determined when the <code>XPathFactory</code> was instantiated.</p>
    *
    * @return New instance of an <code>XPath</code>.
    */
    public abstract XPath newXPath();
}
```

35.8 XPathFactoryConfigurationException.java

Secuencia 326: javax/xml/xpath/XPathFactoryConfigurationException.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

/**
 * <code>XPathFactoryConfigurationException</code> represents a configuration error in a <code>
     XPathFactory</code> environment.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */
public class XPathFactoryConfigurationException extends XPathException {

    /**
     * <p>Stream Unique Identifier.</p>
     */
    private static final long serialVersionUID = -1837080260374986980L;

    /**
     * <p>Constructs a new <code>XPathFactoryConfigurationException</code> with the specified
         detail <code>message</code>.</p>
     *
     * <p>The <code>cause</code> is not initialized.</p>
     *
     * <p>If <code>message</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param message The detail message.
     *
     * @throws NullPointerException When <code>message</code> is
     *   <code>null</code>.
     */
    public XPathFactoryConfigurationException(String message) {
        super(message);
    }

    /**
```

```
     * <p>Constructs a new <code>XPathFactoryConfigurationException</code>
     * with the specified <code>cause</code>.</p>
     *
     * <p>If <code>cause</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param cause The cause.
     *
     * @throws NullPointerException if <code>cause</code> is <code>null</code>.
     */
    public XPathFactoryConfigurationException(Throwable cause) {
        super(cause);
    }
}
```

35.9 XPathFactoryFinder.java

Secuencia 327: javax/xml/xpath/XPathFactoryFinder.java

```
/*
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import java.io.File;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.net.URL;
import java.security.AccessControlContext;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.Properties;
```

```
import java.util.ServiceConfigurationError;
import java.util.ServiceLoader;

/**
 * Implementation of {@link XPathFactory#newInstance(String)}.
 *
 * @author <a href="Kohsuke.Kawaguchi@Sun.com">Kohsuke Kawaguchi</a>
 * @version $Revision: 1.7 $, $Date: 2010-11-01 04:36:14 $
 * @since 1.5
 */
class XPathFactoryFinder  {
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xpath.internal";

    private static final SecuritySupport ss = new SecuritySupport() ;
    /** debug support code. */
    private static boolean debug = false;
    static {
        // Use try/catch block to support applets
        try {
            debug = ss.getSystemProperty("jaxp.debug") != null;
        } catch (Exception unused) {
            debug = false;
        }
    }

    /**
     * <p>Cache properties for performance.</p>
     */
    private static final Properties cacheProps = new Properties();

    /**
     * <p>First time requires initialization overhead.</p>
     */
    private volatile static boolean firstTime = true;

    /**
     * <p>Conditional debug printing.</p>
     *
     * @param msg to print
     */
    private static void debugPrintln(String msg) {
        if (debug) {
            System.err.println("JAXP: " + msg);
        }
    }

    /**
     * <p><code>ClassLoader</code> to use to find <code>XPathFactory</code>.</p>
     */
    private final ClassLoader classLoader;

    /**
     * <p>Constructor that specifies <code>ClassLoader</code> to use
```

```
     * to find <code>XPathFactory</code>.</p>
     *
     * @param loader
     *      to be used to load resource and {@link XPathFactory}
     *      implementations during the resolution process.
     *      If this parameter is null, the default system class loader
     *      will be used.
     */
    public XPathFactoryFinder(ClassLoader loader) {
        this.classLoader = loader;
        if( debug ) {
            debugDisplayClassLoader();
        }
    }

    private void debugDisplayClassLoader() {
        try {
            if( classLoader == ss.getContextClassLoader() ) {
                debugPrintln("using thread context class loader ("+classLoader+") for search");
                return;
            }
        } catch( Throwable unused ) {
             // getContextClassLoader() undefined in JDK1.1
        }

        if( classLoader==ClassLoader.getSystemClassLoader() ) {
            debugPrintln("using system class loader ("+classLoader+") for search");
            return;
        }

        debugPrintln("using class loader ("+classLoader+") for search");
    }

    /**
     * <p>Creates a new {@link XPathFactory} object for the specified
     * object model.</p>
     *
     * @param uri
     *       Identifies the underlying object model.
     *
     * @return <code>null</code> if the callee fails to create one.
     *
     * @throws NullPointerException
     *      If the parameter is null.
     */
    public XPathFactory newFactory(String uri) throws XPathFactoryConfigurationException {
        if (uri == null) {
            throw new NullPointerException();
        }
        XPathFactory f = _newFactory(uri);
        if (f != null) {
            debugPrintln("factory '" + f.getClass().getName() + "' was found for " + uri);
        } else {
```

```
            debugPrintln("unable to find a factory for " + uri);
        }
        return f;
    }

    /**
     * <p>Lookup a {@link XPathFactory} for the given object model.</p>
     *
     * @param uri identifies the object model.
     *
     * @return {@link XPathFactory} for the given object model.
     */
    private XPathFactory _newFactory(String uri) throws XPathFactoryConfigurationException {
        XPathFactory xpathFactory = null;

        String propertyName = SERVICE_CLASS.getName() + ":" + uri;

        // system property look up
        try {
            debugPrintln("Looking up system property '"+propertyName+"'" );
            String r = ss.getSystemProperty(propertyName);
            if(r!=null) {
                debugPrintln("The value is '"+r+"'");
                xpathFactory = createInstance(r, true);
                if (xpathFactory != null) {
                    return xpathFactory;
                }
            } else
                debugPrintln("The property is undefined.");
        } catch( Throwable t ) {
            if( debug ) {
                debugPrintln("failed to look up system property '"+propertyName+"'" );
                t.printStackTrace();
            }
        }

        String javah = ss.getSystemProperty( "java.home" );
        String configFile = javah + File.separator +
        "lib" + File.separator + "jaxp.properties";

        // try to read from $java.home/lib/jaxp.properties
        try {
            if(firstTime){
                synchronized(cacheProps){
                    if(firstTime){
                        File f=new File( configFile );
                        firstTime = false;
                        if(ss.doesFileExist(f)){
                            debugPrintln("Read properties file " + f);
                            cacheProps.load(ss.getFileInputStream(f));
                        }
                    }
                }
```

```
            }
            final String factoryClassName = cacheProps.getProperty(propertyName);
            debugPrintln("found " + factoryClassName + " in $java.home/jaxp.properties");

            if (factoryClassName != null) {
                xpathFactory = createInstance(factoryClassName, true);
                if(xpathFactory != null){
                    return xpathFactory;
                }
            }
        } catch (Exception ex) {
            if (debug) {
                ex.printStackTrace();
            }
        }

        // Try with ServiceLoader
        assert xpathFactory == null;
        xpathFactory = findServiceProvider(uri);

        // The following assertion should always be true.
        // Uncomment it, recompile, and run with -ea in case of doubts:
        // assert xpathFactory == null || xpathFactory.isObjectModelSupported(uri);

        if (xpathFactory != null) {
            return xpathFactory;
        }

        // platform default
        if(uri.equals(XPathFactory.DEFAULT_OBJECT_MODEL_URI)) {
            debugPrintln("attempting to use the platform default W3C DOM XPath lib");
            return createInstance("com.sun.org.apache.xpath.internal.jaxp.XPathFactoryImpl", true);
        }

        debugPrintln("all things were tried, but none was found. bailing out.");
        return null;
    }

    /** <p>Create class using appropriate ClassLoader.</p>
     *
     * @param className Name of class to create.
     * @return Created class or <code>null</code>.
     */
    private Class<?> createClass(String className) {
        Class clazz;
        // make sure we have access to restricted packages
        boolean internal = false;
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                internal = true;
            }
        }

```

```
        // use approprite ClassLoader
        try {
            if (classLoader != null && !internal) {
                    clazz = Class.forName(className, false, classLoader);
            } else {
                    clazz = Class.forName(className);
            }
        } catch (Throwable t) {
            if(debug) {
                t.printStackTrace();
            }
            return null;
        }

        return clazz;
    }

    /**
     * <p>Creates an instance of the specified and returns it.</p>
     *
     * @param className
     *      fully qualified class name to be instantiated.
     *
     * @return null
     *      if it fails. Error messages will be printed by this method.
     */
    XPathFactory createInstance( String className )
            throws XPathFactoryConfigurationException
    {
        return createInstance( className, false );
    }

    XPathFactory createInstance( String className, boolean useServicesMechanism  )
            throws XPathFactoryConfigurationException
    {
        XPathFactory xPathFactory = null;

        debugPrintln("createInstance(" + className + ")");

        // get Class from className
        Class<?> clazz = createClass(className);
        if (clazz == null) {
            debugPrintln("failed to getClass(" + className + ")");
            return null;
        }
        debugPrintln("loaded " + className + " from " + which(clazz));

        // instantiate Class as a XPathFactory
        try {
            if (!useServicesMechanism) {
                xPathFactory = newInstanceNoServiceLoader(clazz);
            }
            if (xPathFactory == null) {
```

```
                xPathFactory = (XPathFactory) clazz.newInstance();
            }
        } catch (ClassCastException classCastException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        classCastException.printStackTrace();
                }
                return null;
        } catch (IllegalAccessException illegalAccessException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        illegalAccessException.printStackTrace();
                }
                return null;
        } catch (InstantiationException instantiationException) {
                debugPrintln("could not instantiate " + clazz.getName());
                if (debug) {
                        instantiationException.printStackTrace();
                }
                return null;
        }

        return xPathFactory;
    }
    /**
     * Try to construct using newXPathFactoryNoServiceLoader
     *   method if available.
     */
    private static XPathFactory newInstanceNoServiceLoader(
         Class<?> providerClass
    ) throws XPathFactoryConfigurationException {
        // Retain maximum compatibility if no security manager.
        if (System.getSecurityManager() == null) {
            return null;
        }
        try {
            Method creationMethod =
                    providerClass.getDeclaredMethod(
                        "newXPathFactoryNoServiceLoader"
                    );
            final int modifiers = creationMethod.getModifiers();

            // Do not call "newXPathFactoryNoServiceLoader" if it's
            // not public static.
            if (!Modifier.isStatic(modifiers) || !Modifier.isPublic(modifiers)) {
                return null;
            }

            // Only calls "newXPathFactoryNoServiceLoader" if it's
            // declared to return an instance of XPathFactory.
            final Class<?> returnType = creationMethod.getReturnType();
            if (SERVICE_CLASS.isAssignableFrom(returnType)) {
                return SERVICE_CLASS.cast(creationMethod.invoke(null, (Object[])null));
```

```
            } else {
                // Should not happen since
                // XPathFactoryImpl.newXPathFactoryNoServiceLoader is
                // declared to return XPathFactory.
                throw new ClassCastException(returnType
                            + " cannot be cast to " + SERVICE_CLASS);
            }
        } catch (ClassCastException e) {
            throw new XPathFactoryConfigurationException(e);
        } catch (NoSuchMethodException exc) {
            return null;
        } catch (Exception exc) {
            return null;
        }
    }

    // Call isObjectModelSupportedBy with initial context.
    private boolean isObjectModelSupportedBy(final XPathFactory factory,
            final String objectModel,
            AccessControlContext acc) {
        return AccessController.doPrivileged(new PrivilegedAction<Boolean>() {
                    public Boolean run() {
                        return factory.isObjectModelSupported(objectModel);
                    }
                }, acc);
    }

    /**
     * Finds a service provider subclass of XPathFactory that supports the
     * given object model using the ServiceLoader.
     *
     * @param objectModel URI of object model to support.
     * @return An XPathFactory supporting the specified object model, or null
     *         if none is found.
     * @throws XPathFactoryConfigurationException if a configuration error is found.
     */
    private XPathFactory findServiceProvider(final String objectModel)
            throws XPathFactoryConfigurationException {

        assert objectModel != null;
        // store current context.
        final AccessControlContext acc = AccessController.getContext();
        try {
            return AccessController.doPrivileged(new PrivilegedAction<XPathFactory>() {
                public XPathFactory run() {
                    final ServiceLoader<XPathFactory> loader =
                            ServiceLoader.load(SERVICE_CLASS);
                    for (XPathFactory factory : loader) {
                        // restore initial context to call
                        // factory.isObjectModelSupportedBy
                        if (isObjectModelSupportedBy(factory, objectModel, acc)) {
                            return factory;
                        }
```

```
                    }
                    return null; // no factory found.
                }
            });
        } catch (ServiceConfigurationError error) {
            throw new XPathFactoryConfigurationException(error);
        }
    }

    private static final Class<XPathFactory> SERVICE_CLASS = XPathFactory.class;

    private static String which( Class clazz ) {
        return which( clazz.getName(), clazz.getClassLoader() );
    }

    /**
     * <p>Search the specified classloader for the given classname.</p>
     *
     * @param classname the fully qualified name of the class to search for
     * @param loader the classloader to search
     *
     * @return the source location of the resource, or null if it wasn't found
     */
    private static String which(String classname, ClassLoader loader) {

        String classnameAsResource = classname.replace('.', '/') + ".class";

        if( loader==null )  loader = ClassLoader.getSystemClassLoader();

        //URL it = loader.getResource(classnameAsResource);
        URL it = ss.getResourceAsURL(loader, classnameAsResource);
        if (it != null) {
            return it.toString();
        } else {
            return null;
        }
    }
}
```

35.10 XPathFunction.java

Secuencia 328: javax/xml/xpath/XPathFunction.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import java.util.List;

/**
 * <p><code>XPathFunction</code> provides access to XPath functions.</p>
 *
 * <p>Functions are identified by QName and arity in XPath.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */
public interface XPathFunction {
  /**
   * <p>Evaluate the function with the specified arguments.</p>
   *
   * <p>To the greatest extent possible, side-effects should be avoided in the
   * definition of extension functions. The implementation evaluating an
   * XPath expression is under no obligation to call extension functions in
   * any particular order or any particular number of times.</p>
   *
   * @param args The arguments, <code>null</code> is a valid value.
   *
   * @return The result of evaluating the <code>XPath</code> function as an <code>Object</code>.
   *
   * @throws XPathFunctionException If <code>args</code> cannot be evaluated with this <code>XPath
       </code> function.
   */
  public Object evaluate(List args)
    throws XPathFunctionException;
}
```

35.11 XPathFunctionException.java

Secuencia 329: javax/xml/xpath/XPathFunctionException.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

/**
 * <code>XPathFunctionException</code> represents an error with an XPath function.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */
public class XPathFunctionException extends XPathExpressionException {

    /**
     * <p>Stream Unique Identifier.</p>
     */
    private static final long serialVersionUID = -1837080260374986980L;

    /**
     * <p>Constructs a new <code>XPathFunctionException</code> with the specified detail <code>
         message</code>.</p>
     *
     * <p>The <code>cause</code> is not initialized.</p>
     *
     * <p>If <code>message</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param message The detail message.
     *
     * @throws NullPointerException When <code>message</code> is
     *   <code>null</code>.
     */
    public XPathFunctionException(String message) {
```

```
        super(message);
    }

    /**
     * <p>Constructs a new <code>XPathFunctionException</code> with the specified <code>cause</code>.</p>
     *
     * <p>If <code>cause</code> is <code>null</code>,
     * then a <code>NullPointerException</code> is thrown.</p>
     *
     * @param cause The cause.
     *
     * @throws NullPointerException if <code>cause</code> is <code>null</code>.
     */
    public XPathFunctionException(Throwable cause) {
        super(cause);
    }
}
```

35.12 XPathFunctionResolver.java

Secuencia 330: javax/xml/xpath/XPathFunctionResolver.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import javax.xml.namespace.QName;

/**
```

```
 * <p><code>XPathFunctionResolver</code> provides access to the set of user defined <code>
     XPathFunction</code>s.</p>
 *
 * <p>XPath functions are resolved by name and arity.
 * The resolver is not needed for XPath built-in functions and the resolver
 * <strong><em>cannot</em></strong> be used to override those functions.</p>
 *
 * <p>In particular, the resolver is only called for functions in an another
 * namespace (functions with an explicit prefix). This means that you cannot
 * use the <code>XPathFunctionResolver</code> to implement specifications
 * like <a href="http://www.w3.org/TR/xmldsig-core/">XML-Signature Syntax
 * and Processing</a> which extend the function library of XPath 1.0 in the
 * same namespace. This is a consequence of the design of the resolver.</p>
 *
 * <p>If you wish to implement additional built-in functions, you will have to
 * extend the underlying implementation directly.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @see <a href="http://www.w3.org/TR/xpath#corelib">XML Path Language (XPath) Version 1.0, Core
     Function Library</a>
 * @since 1.5
 */
public interface XPathFunctionResolver {
  /**
   * <p>Find a function in the set of available functions.</p>
   *
   * <p>If <code>functionName</code> or <code>arity</code> is <code>null</code>, then a <code>
       NullPointerException</code> is thrown.</p>
   *
   * @param functionName The function name.
   * @param arity The number of arguments that the returned function must accept.
   *
   * @return The function or <code>null</code> if no function named <code>functionName</code> with
       <code>arity</code> arguments exists.
   *
   * @throws NullPointerException If <code>functionName</code> or <code>arity</code> is <code>null
       </code>.
   */
  public XPathFunction resolveFunction(QName functionName, int arity);
}
```

35.13 XPathVariableResolver.java

Secuencia 331: javax/xml/xpath/XPathVariableResolver.java

```
/*
 * Copyright (c) 2003, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package javax.xml.xpath;

import javax.xml.namespace.QName;

/**
 * <p><code>XPathVariableResolver</code> provides access to the set of user defined XPath variables
     .</p>
 *
 * <p>The <code>XPathVariableResolver</code> and the XPath evaluator must adhere to a contract that
 * cannot be directly enforced by the API.  Although variables may be mutable,
 * that is, an application may wish to evaluate the same XPath expression more
 * than once with different variable values, in the course of evaluating any
 * single XPath expression, a variable's value <strong><em>must</em></strong>
 * not change.</p>
 *
 * @author  <a href="mailto:Norman.Walsh@Sun.com">Norman Walsh</a>
 * @author  <a href="mailto:Jeff.Suttor@Sun.com">Jeff Suttor</a>
 * @since 1.5
 */
public interface XPathVariableResolver {
  /**
   * <p>Find a variable in the set of available variables.</p>
   *
   * <p>If <code>variableName</code> is <code>null</code>, then a <code>NullPointerException</code>
        is thrown.</p>
   *
   * @param variableName The <code>QName</code> of the variable name.
   *
   * @return The variables value, or <code>null</code> if no variable named <code>variableName</
       code>
   *   exists.  The value returned must be of a type appropriate for the underlying object model.
   *
   * @throws NullPointerException If <code>variableName</code> is <code>null</code>.
   */
  public Object resolveVariable(QName variableName);
}
```

36 Dom (org.w3c.dom package)

36.1 Attr.java

Secuencia 332: org/w3c/dom/Attr.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>Attr</code> interface represents an attribute in an
 * <code>Element</code> object. Typically the allowable values for the
 * attribute are defined in a schema associated with the document.
```

```
* <p><code>Attr</code> objects inherit the <code>Node</code> interface, but
* since they are not actually child nodes of the element they describe, the
* DOM does not consider them part of the document tree. Thus, the
* <code>Node</code> attributes <code>parentNode</code>,
* <code>previousSibling</code>, and <code>nextSibling</code> have a
* <code>null</code> value for <code>Attr</code> objects. The DOM takes the
* view that attributes are properties of elements rather than having a
* separate identity from the elements they are associated with; this should
* make it more efficient to implement such features as default attributes
* associated with all elements of a given type. Furthermore,
* <code>Attr</code> nodes may not be immediate children of a
* <code>DocumentFragment</code>. However, they can be associated with
* <code>Element</code> nodes contained within a
* <code>DocumentFragment</code>. In short, users and implementors of the
* DOM need to be aware that <code>Attr</code> nodes have some things in
* common with other objects inheriting the <code>Node</code> interface, but
* they also are quite distinct.
* <p>The attribute's effective value is determined as follows: if this
* attribute has been explicitly assigned any value, that value is the
* attribute's effective value; otherwise, if there is a declaration for
* this attribute, and that declaration includes a default value, then that
* default value is the attribute's effective value; otherwise, the
* attribute does not exist on this element in the structure model until it
* has been explicitly added. Note that the <code>Node.nodeValue</code>
* attribute on the <code>Attr</code> instance can also be used to retrieve
* the string version of the attribute's value(s).
* <p> If the attribute was not explicitly given a value in the instance
* document but has a default value provided by the schema associated with
* the document, an attribute node will be created with
* <code>specified</code> set to <code>false</code>. Removing attribute
* nodes for which a default value is defined in the schema generates a new
* attribute node with the default value and <code>specified</code> set to
* <code>false</code>. If validation occurred while invoking
* <code>Document.normalizeDocument()</code>, attribute nodes with
* <code>specified</code> equals to <code>false</code> are recomputed
* according to the default attribute values provided by the schema. If no
* default value is associate with this attribute in the schema, the
* attribute node is discarded.
* <p>In XML, where the value of an attribute can contain entity references,
* the child nodes of the <code>Attr</code> node may be either
* <code>Text</code> or <code>EntityReference</code> nodes (when these are
* in use; see the description of <code>EntityReference</code> for
* discussion).
* <p>The DOM Core represents all attribute values as simple strings, even if
* the DTD or schema associated with the document declares them of some
* specific type such as tokenized.
* <p>The way attribute value normalization is performed by the DOM
* implementation depends on how much the implementation knows about the
* schema in use. Typically, the <code>value</code> and
* <code>nodeValue</code> attributes of an <code>Attr</code> node initially
* returns the normalized value given by the parser. It is also the case
* after <code>Document.normalizeDocument()</code> is called (assuming the
* right options have been set). But this may not be the case after
```

```
* mutation, independently of whether the mutation is performed by setting
* the string value directly or by changing the <code>Attr</code> child
* nodes. In particular, this is true when <a href='http://www.w3.org/TR/2004/REC-xml-20040204#dt-charref'>character
* references</a> are involved, given that they are not represented in the DOM and they
* impact attribute value normalization. On the other hand, if the
* implementation knows about the schema in use when the attribute value is
* changed, and it is of a different type than CDATA, it may normalize it
* again at that time. This is especially true of specialized DOM
* implementations, such as SVG DOM implementations, which store attribute
* values in an internal form different from a string.
* <p>The following table gives some examples of the relations between the
* attribute value in the original document (parsed attribute), the value as
* exposed in the DOM, and the serialization of the value:
* <table border='1' cellpadding='3'>
* <tr>
* <th>Examples</th>
* <th>Parsed
* attribute value</th>
* <th>Initial <code>Attr.value</code></th>
* <th>Serialized attribute value</th>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>
* Character reference</td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"x&amp;#178;=5"</pre>
* </td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"x\u00b2=5"</pre>
* </td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"x&amp;#178;=5"</pre>
* </td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>Built-in
* character entity</td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"y&amp;lt;6"</pre>
* </td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"y&lt;6"</pre>
* </td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>"y&amp;lt;6"</pre>
* </td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>Literal newline between</td>
* <td valign='top' rowspan='1' colspan='1'>
* <pre>
* "x=5&amp;#10;y=6"</pre>
```

```
 * </td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>"x=5 y=6"</pre>
 * </td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>"x=5&amp;#10;y=6"</pre>
 * </td>
 * </tr>
 * <tr>
 * <td valign='top' rowspan='1' colspan='1'>Normalized newline between</td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>"x=5
 * y=6"</pre>
 * </td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>"x=5 y=6"</pre>
 * </td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>"x=5 y=6"</pre>
 * </td>
 * </tr>
 * <tr>
 * <td valign='top' rowspan='1' colspan='1'>Entity <code>e</code> with literal newline</td>
 * <td valign='top' rowspan='1' colspan='1'>
 * <pre>
 * &lt;!ENTITY e '...&amp;#10;...'&gt; [...]&gt; "x=5&amp;e;y=6"</pre>
 * </td>
 * <td valign='top' rowspan='1' colspan='1'><em>Dependent on Implementation and Load Options</em></td>
 * <td valign='top' rowspan='1' colspan='1'><em>Dependent on Implementation and Load/Save Options</em></td>
 * </tr>
 * </table>
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Attr extends Node {
    /**
     * Returns the name of this attribute. If <code>Node.localName</code> is
     * different from <code>null</code>, this attribute is a qualified name.
     */
    public String getName();

    /**
     *  <code>True</code> if this attribute was explicitly given a value in
     * the instance document, <code>false</code> otherwise. If the
     * application changed the value of this attribute node (even if it ends
     * up having the same value as the default value) then it is set to
     * <code>true</code>. The implementation may handle attributes with
     * default values from other schemas similarly but applications should
     * use <code>Document.normalizeDocument()</code> to guarantee this
     * information is up-to-date.
     */
```

```
    public boolean getSpecified();

    /**
     * On retrieval, the value of the attribute is returned as a string.
     * Character and general entity references are replaced with their
     * values. See also the method <code>getAttribute</code> on the
     * <code>Element</code> interface.
     * <br>On setting, this creates a <code>Text</code> node with the unparsed
     * contents of the string, i.e. any characters that an XML processor
     * would recognize as markup are instead treated as literal text. See
     * also the method <code>Element.setAttribute()</code>.
     * <br> Some specialized implementations, such as some [<a href='http://www.w3.org/TR/2003/REC-
         SVG11-20030114/'>SVG 1.1</a>]
     * implementations, may do normalization automatically, even after
     * mutation; in such case, the value on retrieval may differ from the
     * value on setting.
     */
    public String getValue();
    /**
     * On retrieval, the value of the attribute is returned as a string.
     * Character and general entity references are replaced with their
     * values. See also the method <code>getAttribute</code> on the
     * <code>Element</code> interface.
     * <br>On setting, this creates a <code>Text</code> node with the unparsed
     * contents of the string, i.e. any characters that an XML processor
     * would recognize as markup are instead treated as literal text. See
     * also the method <code>Element.setAttribute()</code>.
     * <br> Some specialized implementations, such as some [<a href='http://www.w3.org/TR/2003/REC-
         SVG11-20030114/'>SVG 1.1</a>]
     * implementations, may do normalization automatically, even after
     * mutation; in such case, the value on retrieval may differ from the
     * value on setting.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.
     */
    public void setValue(String value)
                            throws DOMException;

    /**
     * The <code>Element</code> node this attribute is attached to or
     * <code>null</code> if this attribute is not in use.
     * @since DOM Level 2
     */
    public Element getOwnerElement();

    /**
     *  The type information associated with this attribute. While the type
     * information contained in this attribute is guarantee to be correct
     * after loading the document or invoking
     * <code>Document.normalizeDocument()</code>, <code>schemaTypeInfo</code>
     *  may not be reliable if the node was moved.
     * @since DOM Level 3
     */
```

```
    public TypeInfo getSchemaTypeInfo();

    /**
     *  Returns whether this attribute is known to be of type ID (i.e. to
     * contain an identifier for its owner element) or not. When it is and
     * its value is unique, the <code>ownerElement</code> of this attribute
     * can be retrieved using the method <code>Document.getElementById</code>
     * . The implementation could use several ways to determine if an
     * attribute node is known to contain an identifier:
     * <ul>
     * <li> If validation
     * occurred using an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>
         XML Schema Part 1</a>]
     *  while loading the document or while invoking
     * <code>Document.normalizeDocument()</code>, the post-schema-validation
     * infoset contributions (PSVI contributions) values are used to
     * determine if this attribute is a schema-determined ID attribute using
     * the <a href='http://www.w3.org/TR/2003/REC-xptr-framework-20030325/#term-sdi'>
     * schema-determined ID</a> definition in [<a href='http://www.w3.org/TR/2003/REC-xptr-
         framework-20030325/'>XPointer</a>]
     * .
     * </li>
     * <li> If validation occurred using a DTD while loading the document or
     * while invoking <code>Document.normalizeDocument()</code>, the infoset <b>[type definition]</
         b> value is used to determine if this attribute is a DTD-determined ID
     * attribute using the <a href='http://www.w3.org/TR/2003/REC-xptr-framework-20030325/#term-ddi
         '>
     * DTD-determined ID</a> definition in [<a href='http://www.w3.org/TR/2003/REC-xptr-framework
         -20030325/'>XPointer</a>]
     * .
     * </li>
     * <li> from the use of the methods <code>Element.setIdAttribute()</code>,
     * <code>Element.setIdAttributeNS()</code>, or
     * <code>Element.setIdAttributeNode()</code>, i.e. it is an
     * user-determined ID attribute;
     * <p ><b>Note:</b>  XPointer framework (see section 3.2 in [<a href='http://www.w3.org/TR
         /2003/REC-xptr-framework-20030325/'>XPointer</a>]
     * ) consider the DOM user-determined ID attribute as being part of the
     * XPointer externally-determined ID definition.
     * </li>
     * <li> using mechanisms that
     * are outside the scope of this specification, it is then an
     * externally-determined ID attribute. This includes using schema
     * languages different from XML schema and DTD.
     * </li>
     * </ul>
     * <br> If validation occurred while invoking
     * <code>Document.normalizeDocument()</code>, all user-determined ID
     * attributes are reset and all attribute nodes ID information are then
     * reevaluated in accordance to the schema used. As a consequence, if
     * the <code>Attr.schemaTypeInfo</code> attribute contains an ID type,
     * <code>isId</code> will always return true.
     * @since DOM Level 3
```

```
     */
    public boolean isId();

}
```

36.2 CDATASection.java

Secuencia 333: org/w3c/dom/CDATASection.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * CDATA sections are used to escape blocks of text containing characters that
```

```
 * would otherwise be regarded as markup. The only delimiter that is
 * recognized in a CDATA section is the "]]&gt;" string that ends the CDATA
 * section. CDATA sections cannot be nested. Their primary purpose is for
 * including material such as XML fragments, without needing to escape all
 * the delimiters.
 * <p>The <code>CharacterData.data</code> attribute holds the text that is
 * contained by the CDATA section. Note that this <em>may</em> contain characters that need to be escaped outside of CDATA sections and
 * that, depending on the character encoding ("charset") chosen for
 * serialization, it may be impossible to write out some characters as part
 * of a CDATA section.
 * <p>The <code>CDATASection</code> interface inherits from the
 * <code>CharacterData</code> interface through the <code>Text</code>
 * interface. Adjacent <code>CDATASection</code> nodes are not merged by use
 * of the <code>normalize</code> method of the <code>Node</code> interface.
 * <p> No lexical check is done on the content of a CDATA section and it is
 * therefore possible to have the character sequence <code>"]]&gt;"</code>
 * in the content, which is illegal in a CDATA section per section 2.7 of [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]. The
 * presence of this character sequence must generate a fatal error during
 * serialization or the cdata section must be splitted before the
 * serialization (see also the parameter <code>"split-cdata-sections"</code>
 * in the <code>DOMConfiguration</code> interface).
 * <p ><b>Note:</b> Because no markup is recognized within a
 * <code>CDATASection</code>, character numeric references cannot be used as
 * an escape mechanism when serializing. Therefore, action needs to be taken
 * when serializing a <code>CDATASection</code> with a character encoding
 * where some of the contained characters cannot be represented. Failure to
 * do so would not produce well-formed XML.
 * <p ><b>Note:</b> One potential solution in the serialization process is to
 * end the CDATA section before the character, output the character using a
 * character reference or entity reference, and open a new CDATA section for
 * any further characters in the text node. Note, however, that some code
 * conversion libraries at the time of writing do not return an error or
 * exception when a character is missing from the encoding, making the task
 * of ensuring that data is not corrupted on serialization more difficult.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface CDATASection extends Text {
}
```

36.3 CharacterData.java

Secuencia 334: org/w3c/dom/CharacterData.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>CharacterData</code> interface extends Node with a set of
 * attributes and methods for accessing character data in the DOM. For
 * clarity this set is defined here rather than on each object that uses
 * these attributes and methods. No DOM objects correspond directly to
 * <code>CharacterData</code>, though <code>Text</code> and others do
 * inherit the interface from it. All <code>offsets</code> in this interface
 * start from <code>0</code>.
 * <p>As explained in the <code>DOMString</code> interface, text strings in
 * the DOM are represented in UTF-16, i.e. as a sequence of 16-bit units. In
 * the following, the term 16-bit units is used whenever necessary to
 * indicate that indexing on CharacterData is done in 16-bit units.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface CharacterData extends Node {
    /**
     * The character data of the node that implements this interface. The DOM
```

```
     * implementation may not put arbitrary limits on the amount of data
     * that may be stored in a <code>CharacterData</code> node. However,
     * implementation limits may mean that the entirety of a node's data may
     * not fit into a single <code>DOMString</code>. In such cases, the user
     * may call <code>substringData</code> to retrieve the data in
     * appropriately sized pieces.
     * @exception DOMException
     *   DOMSTRING_SIZE_ERR: Raised when it would return more characters than
     *   fit in a <code>DOMString</code> variable on the implementation
     *   platform.
     */
    public String getData()
                            throws DOMException;
    /**
     * The character data of the node that implements this interface. The DOM
     * implementation may not put arbitrary limits on the amount of data
     * that may be stored in a <code>CharacterData</code> node. However,
     * implementation limits may mean that the entirety of a node's data may
     * not fit into a single <code>DOMString</code>. In such cases, the user
     * may call <code>substringData</code> to retrieve the data in
     * appropriately sized pieces.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.
     */
    public void setData(String data)
                            throws DOMException;

    /**
     * The number of 16-bit units that are available through <code>data</code>
     * and the <code>substringData</code> method below. This may have the
     * value zero, i.e., <code>CharacterData</code> nodes may be empty.
     */
    public int getLength();

    /**
     * Extracts a range of data from the node.
     * @param offset Start offset of substring to extract.
     * @param count The number of 16-bit units to extract.
     * @return The specified substring. If the sum of <code>offset</code> and
     *   <code>count</code> exceeds the <code>length</code>, then all 16-bit
     *   units to the end of the data are returned.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified <code>offset</code> is
     *   negative or greater than the number of 16-bit units in
     *   <code>data</code>, or if the specified <code>count</code> is
     *   negative.
     *   <br>DOMSTRING_SIZE_ERR: Raised if the specified range of text does
     *   not fit into a <code>DOMString</code>.
     */
    public String substringData(int offset,
                                int count)
                                throws DOMException;

```

```
    /**
     * Append the string to the end of the character data of the node. Upon
     * success, <code>data</code> provides access to the concatenation of
     * <code>data</code> and the <code>DOMString</code> specified.
     * @param arg The <code>DOMString</code> to append.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void appendData(String arg)
                           throws DOMException;

    /**
     * Insert a string at the specified 16-bit unit offset.
     * @param offset The character offset at which to insert.
     * @param arg The <code>DOMString</code> to insert.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified <code>offset</code> is
     *   negative or greater than the number of 16-bit units in
     *   <code>data</code>.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void insertData(int offset,
                           String arg)
                           throws DOMException;

    /**
     * Remove a range of 16-bit units from the node. Upon success,
     * <code>data</code> and <code>length</code> reflect the change.
     * @param offset The offset from which to start removing.
     * @param count The number of 16-bit units to delete. If the sum of
     *   <code>offset</code> and <code>count</code> exceeds
     *   <code>length</code> then all 16-bit units from <code>offset</code>
     *   to the end of the data are deleted.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified <code>offset</code> is
     *   negative or greater than the number of 16-bit units in
     *   <code>data</code>, or if the specified <code>count</code> is
     *   negative.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void deleteData(int offset,
                           int count)
                           throws DOMException;

    /**
     * Replace the characters starting at the specified 16-bit unit offset
     * with the specified string.
     * @param offset The offset from which to start replacing.
     * @param count The number of 16-bit units to replace. If the sum of
     *   <code>offset</code> and <code>count</code> exceeds
     *   <code>length</code>, then all 16-bit units to the end of the data
     *   are replaced; (i.e., the effect is the same as a <code>remove</code>
     *    method call with the same range, followed by an <code>append</code>
```

```
     *    method invocation).
     * @param arg The <code>DOMString</code> with which the range must be
     *   replaced.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified <code>offset</code> is
     *   negative or greater than the number of 16-bit units in
     *   <code>data</code>, or if the specified <code>count</code> is
     *   negative.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void replaceData(int offset,
                            int count,
                            String arg)
                            throws DOMException;

}
```

36.4 Comment.java

Secuencia 335: org/w3c/dom/Comment.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
```

```
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * This interface inherits from <code>CharacterData</code> and represents the
 * content of a comment, i.e., all the characters between the starting '
 * <code>&lt;!--</code>' and ending '<code>--&gt;</code>'. Note that this is
 * the definition of a comment in XML, and, in practice, HTML, although some
 * HTML tools may implement the full SGML comment structure.
 * <p> No lexical check is done on the content of a comment and it is
 * therefore possible to have the character sequence <code>"--"</code>
 * (double-hyphen) in the content, which is illegal in a comment per section
 * 2.5 of [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]. The
 * presence of this character sequence must generate a fatal error during
 * serialization.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Comment extends CharacterData {
}
```

36.5 DOMConfiguration.java

Secuencia 336: org/w3c/dom/DOMConfiguration.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  The <code>DOMConfiguration</code> interface represents the configuration
 * of a document and maintains a table of recognized parameters. Using the
 * configuration, it is possible to change
 * <code>Document.normalizeDocument()</code> behavior, such as replacing the
 * <code>CDATASection</code> nodes with <code>Text</code> nodes or
 * specifying the type of the schema that must be used when the validation
 * of the <code>Document</code> is requested. <code>DOMConfiguration</code>
 * objects are also used in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>DOM Level 3 Load and Save</a>]
 *  in the <code>DOMParser</code> and <code>DOMSerializer</code> interfaces.
 * <p> The parameter names used by the <code>DOMConfiguration</code> object
 * are defined throughout the DOM Level 3 specifications. Names are
 * case-insensitive. To avoid possible conflicts, as a convention, names
 * referring to parameters defined outside the DOM specification should be
 * made unique. Because parameters are exposed as properties in names
 * are recommended to follow the section 5.16 Identifiers of [Unicode] with the addition of the character '-' (HYPHEN-MINUS) but it is not
 * enforced by the DOM implementation. DOM Level 3 Core Implementations are
 * required to recognize all parameters defined in this specification. Some
 * parameter values may also be required to be supported by the
 * implementation. Refer to the definition of the parameter to know if a
 * value must be supported or not.
 * <p ><b>Note:</b>  Parameters are similar to features and properties used in
 * SAX2 [<a href='http://www.saxproject.org/'>SAX</a>].
 * <p> The following list of parameters defined in the DOM:
 * <dl>
 * <dt>
 * <code>"canonical-form"</code></dt>
 * <dd>
 * <dl>
 * <dt><code>true</code></dt>
```

```
* <dd>[<em>optional</em>] Canonicalize the document according to the rules specified in [<a href='http://www.w3.org/TR/2001/REC-xml-c14n-20010315'>Canonical XML</a>],
* such as removing the <code>DocumentType</code> node (if any) from the
* tree, or removing superfluous namespace declarations from each element.
* Note that this is limited to what can be represented in the DOM; in
* particular, there is no way to specify the order of the attributes in the
* DOM. In addition,  Setting this parameter to <code>true</code> will also
* set the state of the parameters listed below. Later changes to the state
* of one of those parameters will revert "canonical-form" back to
* <code>false</code>. Parameters set to <code>false</code>: "entities", "
* normalize-characters", "cdata-sections". Parameters set to
* <code>true</code>: "namespaces", "namespace-declarations", "well-formed",
* "element-content-whitespace". Other parameters are not changed unless
* explicitly specified in the description of the parameters.</dd>
* <dt>
* <code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Do not canonicalize the document.</dd>
* </dl></dd>
* <dt><code>"cdata-sections"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Keep <code>CDATASection</code> nodes in the document.</dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>]Transform <code>CDATASection</code> nodes in the document into
* <code>Text</code> nodes. The new <code>Text</code> node is then combined
* with any adjacent <code>Text</code> node.</dd>
* </dl></dd>
* <dt>
* <code>"check-character-normalization"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>optional</em>] Check if the characters in the document are <a href='http://www.w3.org/TR/2004/REC-xml11-20040204/#dt-fullnorm'>fully
* normalized</a>, as defined in appendix B of [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>]. When a
* sequence of characters is encountered that fails normalization checking,
* an error with the <code>DOMError.type</code> equals to
* "check-character-normalization-failure" is issued. </dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Do not check if characters are normalized.</dd>
* </dl></dd>
* <dt><code>"comments"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Keep <code>Comment</code> nodes in the document.</dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>]Discard <code>Comment</code> nodes in the document.</dd>
```

```
* </dl></dd>
* <dt>
* <code>"datatype-normalization"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>optional</em>] Expose schema normalized values in the tree, such as <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#key-nv'>XML
* Schema normalized values</a> in the case of XML Schema. Since this parameter requires to have schema
* information, the "validate" parameter will also be set to
* <code>true</code>. Having this parameter activated when "validate" is
* <code>false</code> has no effect and no schema-normalization will happen.
* <p ><b>Note:</b>  Since the document contains the result of the XML 1.0
* processing, this parameter does not apply to attribute value
* normalization as defined in section 3.3.3 of [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>] and is only
* meant for schema languages other than Document Type Definition (DTD). </dd>
* <dt>
* <code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) Do not perform schema normalization on the tree. </dd>
* </dl></dd>
* <dt>
* <code>"element-content-whitespace"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Keep all whitespaces in the document.</dd>
* <dt><code>false</code></dt>
* <dd>[<em>optional</em>] Discard all <code>Text</code> nodes that contain whitespaces in element
* content, as described in <a href='http://www.w3.org/TR/2004/REC-xml-infoset-20040204#infoitem.character'>
* [element content whitespace]</a>. The implementation is expected to use the attribute
* <code>Text.isElementContentWhitespace</code> to determine if a
* <code>Text</code> node should be discarded or not.</dd>
* </dl></dd>
* <dt><code>"entities"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Keep <code>EntityReference</code> nodes in the document.</dd>
* <dt>
* <code>false</code></dt>
* <dd>[<em>required</em>] Remove all <code>EntityReference</code> nodes from the document,
* putting the entity expansions directly in their place. <code>Text</code>
* nodes are normalized, as defined in <code>Node.normalize</code>. Only <a href='http://www.w3.org/TR/2004/REC-xml-infoset-20040204/#infoitem.rse'>
* unexpanded entity references</a> are kept in the document. </dd>
* </dl>
* <p ><b>Note:</b>  This parameter does not affect <code>Entity</code> nodes. </dd>
```

```
* <dt>
* <code>"error-handler"</code></dt>
* <dd>[<em>required</em>] Contains a <code>DOMErrorHandler</code> object. If an error is
* encountered in the document, the implementation will call back the
* <code>DOMErrorHandler</code> registered using this parameter. The
* implementation may provide a default <code>DOMErrorHandler</code> object.
*  When called, <code>DOMError.relatedData</code> will contain the closest
* node to where the error occurred. If the implementation is unable to
* determine the node where the error occurs,
* <code>DOMError.relatedData</code> will contain the <code>Document</code>
* node. Mutations to the document from within an error handler will result
* in implementation dependent behavior. </dd>
* <dt><code>"infoset"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>]Keep in the document the information defined in the XML Information Set
    [<a href='http://www.w3.org/TR/2004/REC-xml-infoset-20040204/'>XML Information Set</a>]
* .This forces the following parameters to <code>false</code>: "
* validate-if-schema", "entities", "datatype-normalization", "cdata-sections
* ".This forces the following parameters to <code>true</code>: "
* namespace-declarations", "well-formed", "element-content-whitespace", "
* comments", "namespaces".Other parameters are not changed unless
* explicitly specified in the description of the parameters. Note that
* querying this parameter with <code>getParameter</code> returns
* <code>true</code> only if the individual parameters specified above are
* appropriately set.</dd>
* <dt><code>false</code></dt>
* <dd>Setting <code>infoset</code> to
* <code>false</code> has no effect.</dd>
* </dl></dd>
* <dt><code>"namespaces"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) Perform the namespace processing as defined in . </dd
    >
* <dt><code>false</code></dt>
* <dd>[<em>optional</em>] Do not perform the namespace processing. </dd>
* </dl></dd>
* <dt>
* <code>"namespace-declarations"</code></dt>
* <dd> This parameter has no effect if the
* parameter "namespaces" is set to <code>false</code>.
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) Include namespace declaration attributes, specified
    or defaulted from
* the schema, in the document. See also the sections "Declaring Namespaces"
* in [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
*  and [<a href='http://www.w3.org/TR/2004/REC-xml-names11-20040204/'>XML Namespaces 1.1</a>]
```

```
* .</dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>]Discard all namespace declaration attributes. The namespace prefixes (
* <code>Node.prefix</code>) are retained even if this parameter is set to
* <code>false</code>.</dd>
* </dl></dd>
* <dt><code>"normalize-characters"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>optional</em>] <a href='http://www.w3.org/TR/2004/REC-xml11-20040204/#dt-fullnorm'>Fully
* normalized</a> the characters in the document as defined in appendix B of [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>]. </dd>
* <dt>
* <code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Do not perform character normalization.</dd>
* </dl></dd>
* <dt><code>"schema-location"</code></dt>
* <dd>[<em>optional</em>] Represent a <code>DOMString</code> object containing a list of URIs,
* separated by whitespaces (characters matching the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#NT-S'>nonterminal
* production S</a> defined in section 2.3 [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]), that
* represents the schemas against which validation should occur, i.e. the
* current schema. The types of schemas referenced in this list must match
* the type specified with <code>schema-type</code>, otherwise the behavior
* of an implementation is undefined.  The schemas specified using this
* property take precedence to the schema information specified in the
* document itself. For namespace aware schema, if a schema specified using
* this property and a schema specified in the document instance (i.e. using
* the <code>schemaLocation</code> attribute) in a schema document (i.e.
* using schema <code>import</code> mechanisms) share the same
* <code>targetNamespace</code>, the schema specified by the user using this
* property will be used. If two schemas specified using this property share
* the same <code>targetNamespace</code> or have no namespace, the behavior
* is implementation dependent.  If no location has been provided, this
* parameter is <code>null</code>.
* <p ><b>Note:</b>  The <code>"schema-location"</code> parameter is ignored
* unless the "schema-type" parameter value is set. It is strongly
* recommended that <code>Document.documentURI</code> will be set so that an
* implementation can successfully resolve any external entities referenced. </dd>
* <dt>
* <code>"schema-type"</code></dt>
* <dd>[<em>optional</em>] Represent a <code>DOMString</code> object containing an absolute URI
* and representing the type of the schema language used to validate a
* document against. Note that no lexical checking is done on the absolute
* URI.  If this parameter is not set, a default value may be provided by
* the implementation, based on the schema languages supported and on the
* schema language used at load time. If no value is provided, this
* parameter is <code>null</code>.
* <p ><b>Note:</b>  For XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
```

```
* , applications must use the value
* <code>"http://www.w3.org/2001/XMLSchema"</code>. For XML DTD [<a href='http://www.w3.org/TR
    /2004/REC-xml-20040204'>XML 1.0</a>],
* applications must use the value
* <code>"http://www.w3.org/TR/REC-xml"</code>. Other schema languages are
* outside the scope of the W3C and therefore should recommend an absolute
* URI in order to use this method. </dd>
* <dt><code>"split-cdata-sections"</code></dt>
* <dd>
* <dl>
* <dt>
* <code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>)Split CDATA sections containing the CDATA section
    termination marker
* ']]&gt;'. When a CDATA section is split a warning is issued with a
* <code>DOMError.type</code> equals to
* <code>"cdata-sections-splitted"</code> and
* <code>DOMError.relatedData</code> equals to the first
* <code>CDATASection</code> node in document order resulting from the split.</dd>
* <dt>
* <code>false</code></dt>
* <dd>[<em>required</em>]Signal an error if a <code>CDATASection</code> contains an
* unrepresentable character.</dd>
* </dl></dd>
* <dt><code>"validate"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>optional</em>] Require the validation against a schema (i.e. XML schema, DTD, any
* other type or representation of schema) of the document as it is being
* normalized as defined by [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]. If
* validation errors are found, or no schema was found, the error handler is
* notified. Schema-normalized values will not be exposed according to the
* schema in used unless the parameter "datatype-normalization" is
* <code>true</code>.  This parameter will reevaluate:
* <ul>
* <li> Attribute nodes with
* <code>Attr.specified</code> equals to <code>false</code>, as specified in
* the description of the <code>Attr</code> interface;
* </li>
* <li> The value of the
* attribute <code>Text.isElementContentWhitespace</code> for all
* <code>Text</code> nodes;
* </li>
* <li> The value of the attribute
* <code>Attr.isId</code> for all <code>Attr</code> nodes;
* </li>
* <li> The attributes
* <code>Element.schemaTypeInfo</code> and <code>Attr.schemaTypeInfo</code>.
* </li>
* </ul>
* <p ><b>Note:</b>  "validate-if-schema" and "validate" are mutually
* exclusive, setting one of them to <code>true</code> will set the other
```

```
* one to <code>false</code>. Applications should also consider setting the
* parameter "well-formed" to <code>true</code>, which is the default for
* that option, when validating the document. </dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) Do not accomplish schema processing, including the internal subset
* processing. Default attribute values information are kept. Note that
* validation might still happen if "validate-if-schema" is <code>true</code>
* . </dd>
* </dl></dd>
* <dt><code>"validate-if-schema"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>optional</em>]Enable validation only if a declaration for the document element can be
* found in a schema (independently of where it is found, i.e. XML schema,
* DTD, or any other type or representation of schema). If validation is
* enabled, this parameter has the same behavior as the parameter "validate"
* set to <code>true</code>.
* <p ><b>Note:</b>  "validate-if-schema" and "validate" are mutually
* exclusive, setting one of them to <code>true</code> will set the other
* one to <code>false</code>. </dd>
* <dt><code>false</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) No schema processing should be performed if the document has a schema,
* including internal subset processing. Default attribute values
* information are kept. Note that validation must still happen if "validate
* " is <code>true</code>. </dd>
* </dl></dd>
* <dt><code>"well-formed"</code></dt>
* <dd>
* <dl>
* <dt><code>true</code></dt>
* <dd>[<em>required</em>] (<em>default</em>) Check if all nodes are XML well formed according to the XML version in
* use in <code>Document.xmlVersion</code>:
* <ul>
* <li> check if the attribute
* <code>Node.nodeName</code> contains invalid characters according to its
* node type and generate a <code>DOMError</code> of type
* <code>"wf-invalid-character-in-node-name"</code>, with a
* <code>DOMError.SEVERITY_ERROR</code> severity, if necessary;
* </li>
* <li> check if
* the text content inside <code>Attr</code>, <code>Element</code>,
* <code>Comment</code>, <code>Text</code>, <code>CDATASection</code> nodes
* for invalid characters and generate a <code>DOMError</code> of type
* <code>"wf-invalid-character"</code>, with a
* <code>DOMError.SEVERITY_ERROR</code> severity, if necessary;
* </li>
* <li> check if
* the data inside <code>ProcessingInstruction</code> nodes for invalid
* characters and generate a <code>DOMError</code> of type
```

```
 * <code>"wf-invalid-character"</code>, with a
 * <code>DOMError.SEVERITY_ERROR</code> severity, if necessary;
 * </li>
 * </ul></dd>
 * <dt>
 * <code>false</code></dt>
 * <dd>[<em>optional</em>] Do not check for XML well-formedness. </dd>
 * </dl></dd>
 * </dl>
 * <p> The resolution of the system identifiers associated with entities is
 * done using <code>Document.documentURI</code>. However, when the feature
 * "LS" defined in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>DOM Level 3
     Load and Save</a>]
 *  is supported by the DOM implementation, the parameter
 * "resource-resolver" can also be used on <code>DOMConfiguration</code>
 * objects attached to <code>Document</code> nodes. If this parameter is
 * set, <code>Document.normalizeDocument()</code> will invoke the resource
 * resolver instead of using <code>Document.documentURI</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMConfiguration {
    /**
     * Set the value of a parameter.
     * @param name The name of the parameter to set.
     * @param value  The new value or <code>null</code> if the user wishes to
     *   unset the parameter. While the type of the value parameter is
     *   defined as <code>DOMUserData</code>, the object type must match the
     *   type defined by the definition of the parameter. For example, if
     *   the parameter is "error-handler", the value must be of type
     *   <code>DOMErrorHandler</code>.
     * @exception DOMException
     *    NOT_FOUND_ERR: Raised when the parameter name is not recognized.
     *   <br> NOT_SUPPORTED_ERR: Raised when the parameter name is recognized
     *   but the requested value cannot be set.
     *   <br> TYPE_MISMATCH_ERR: Raised if the value type for this parameter
     *   name is incompatible with the expected value type.
     */
    public void setParameter(String name,
                             Object value)
                             throws DOMException;

    /**
     *  Return the value of a parameter if known.
     * @param name  The name of the parameter.
     * @return  The current object associated with the specified parameter or
     *   <code>null</code> if no object has been associated or if the
     *   parameter is not supported.
     * @exception DOMException
     *    NOT_FOUND_ERR: Raised when the parameter name is not recognized.
     */
    public Object getParameter(String name)
```

```
                                throws DOMException;

    /**
     * Check if setting a parameter to a specific value is supported.
     * @param name The name of the parameter to check.
     * @param value  An object. if <code>null</code>, the returned value is
     *   <code>true</code>.
     * @return  <code>true</code> if the parameter could be successfully set
     *   to the specified value, or <code>false</code> if the parameter is
     *   not recognized or the requested value is not supported. This does
     *   not change the current value of the parameter itself.
     */
    public boolean canSetParameter(String name,
                                   Object value);

    /**
     *  The list of the parameters supported by this
     * <code>DOMConfiguration</code> object and for which at least one value
     * can be set by the application. Note that this list can also contain
     * parameter names defined outside this specification.
     */
    public DOMStringList getParameterNames();

}
```

36.6 DOMError.java

Secuencia 337: org/w3c/dom/DOMError.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
```

```
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * <code>DOMError</code> is an interface that describes an error.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMError {
    // ErrorSeverity
    /**
     * The severity of the error described by the <code>DOMError</code> is
     * warning. A <code>SEVERITY_WARNING</code> will not cause the
     * processing to stop, unless <code>DOMErrorHandler.handleError()</code>
     * returns <code>false</code>.
     */
    public static final short SEVERITY_WARNING          = 1;
    /**
     * The severity of the error described by the <code>DOMError</code> is
     * error. A <code>SEVERITY_ERROR</code> may not cause the processing to
     * stop if the error can be recovered, unless
     * <code>DOMErrorHandler.handleError()</code> returns <code>false</code>.
     */
    public static final short SEVERITY_ERROR            = 2;
    /**
     * The severity of the error described by the <code>DOMError</code> is
     * fatal error. A <code>SEVERITY_FATAL_ERROR</code> will cause the
     * normal processing to stop. The return value of
     * <code>DOMErrorHandler.handleError()</code> is ignored unless the
     * implementation chooses to continue, in which case the behavior
     * becomes undefined.
     */
    public static final short SEVERITY_FATAL_ERROR      = 3;

    /**
     * The severity of the error, either <code>SEVERITY_WARNING</code>,
     * <code>SEVERITY_ERROR</code>, or <code>SEVERITY_FATAL_ERROR</code>.
```

```
     */
    public short getSeverity();

    /**
     * An implementation specific string describing the error that occurred.
     */
    public String getMessage();

    /**
     *  A <code>DOMString</code> indicating which related data is expected in
     * <code>relatedData</code>. Users should refer to the specification of
     * the error in order to find its <code>DOMString</code> type and
     * <code>relatedData</code> definitions if any.
     * <p ><b>Note:</b>  As an example,
     * <code>Document.normalizeDocument()</code> does generate warnings when
     * the "split-cdata-sections" parameter is in use. Therefore, the method
     * generates a <code>SEVERITY_WARNING</code> with <code>type</code>
     * <code>"cdata-sections-splitted"</code> and the first
     * <code>CDATASection</code> node in document order resulting from the
     * split is returned by the <code>relatedData</code> attribute.
     */
    public String getType();

    /**
     * The related platform dependent exception if any.
     */
    public Object getRelatedException();

    /**
     *  The related <code>DOMError.type</code> dependent data if any.
     */
    public Object getRelatedData();

    /**
     * The location of the error.
     */
    public DOMLocator getLocation();

}
```

36.7 DOMErrorHandler.java

Secuencia 338: org/w3c/dom/DOMErrorHandler.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  <code>DOMErrorHandler</code> is a callback interface that the DOM
 * implementation can call when reporting errors that happens while
 * processing XML data, or when doing some other processing (e.g. validating
 * a document). A <code>DOMErrorHandler</code> object can be attached to a
 * <code>Document</code> using the "error-handler" on the
 * <code>DOMConfiguration</code> interface. If more than one error needs to
 * be reported during an operation, the sequence and numbers of the errors
 * passed to the error handler are implementation dependent.
 * <p> The application that is using the DOM implementation is expected to
 * implement this interface.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMErrorHandler {
    /**
     * This method is called on the error handler when an error occurs.
     * <br> If an exception is thrown from this method, it is considered to be
     * equivalent of returning <code>true</code>.
```

```
     * @param error  The error object that describes the error. This object
     *   may be reused by the DOM implementation across multiple calls to
     *   the <code>handleError</code> method.
     * @return  If the <code>handleError</code> method returns
     *   <code>false</code>, the DOM implementation should stop the current
     *   processing when possible. If the method returns <code>true</code>,
     *   the processing may continue depending on
     *   <code>DOMError.severity</code>.
     */
    public boolean handleError(DOMError error);

}
```

36.8 DOMException.java

Secuencia 339: org/w3c/dom/DOMException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * DOM operations only raise exceptions in "exceptional" circumstances, i.e.,
 * when an operation is impossible to perform (either for logical reasons,
 * because data is lost, or because the implementation has become unstable).
 * In general, DOM methods return specific error values in ordinary
 * processing situations, such as out-of-bound errors when using
 * <code>NodeList</code>.
 * <p>Implementations should raise other exceptions under other circumstances.
 * For example, implementations should raise an implementation-dependent
 * exception if a <code>null</code> argument is passed when <code>null</code>
 *  was not expected.
 * <p>Some languages and object systems do not support the concept of
 * exceptions. For such systems, error conditions may be indicated using
 * native error reporting mechanisms. For some bindings, for example,
 * methods may return error codes similar to those listed in the
 * corresponding method descriptions.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public class DOMException extends RuntimeException {
    public DOMException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
    // ExceptionCode
    /**
     * If index or size is negative, or greater than the allowed value.
     */
    public static final short INDEX_SIZE_ERR            = 1;
    /**
     * If the specified range of text does not fit into a
     * <code>DOMString</code>.
     */
    public static final short DOMSTRING_SIZE_ERR        = 2;
    /**
     * If any <code>Node</code> is inserted somewhere it doesn't belong.
     */
    public static final short HIERARCHY_REQUEST_ERR     = 3;
    /**
     * If a <code>Node</code> is used in a different document than the one
     * that created it (that doesn't support it).
     */
    public static final short WRONG_DOCUMENT_ERR        = 4;
    /**
     * If an invalid or illegal character is specified, such as in an XML name.
     */
```

```
    public static final short INVALID_CHARACTER_ERR     = 5;
    /**
     * If data is specified for a <code>Node</code> which does not support
     * data.
     */
    public static final short NO_DATA_ALLOWED_ERR       = 6;
    /**
     * If an attempt is made to modify an object where modifications are not
     * allowed.
     */
    public static final short NO_MODIFICATION_ALLOWED_ERR = 7;
    /**
     * If an attempt is made to reference a <code>Node</code> in a context
     * where it does not exist.
     */
    public static final short NOT_FOUND_ERR             = 8;
    /**
     * If the implementation does not support the requested type of object or
     * operation.
     */
    public static final short NOT_SUPPORTED_ERR         = 9;
    /**
     * If an attempt is made to add an attribute that is already in use
     * elsewhere.
     */
    public static final short INUSE_ATTRIBUTE_ERR       = 10;
    /**
     * If an attempt is made to use an object that is not, or is no longer,
     * usable.
     * @since DOM Level 2
     */
    public static final short INVALID_STATE_ERR         = 11;
    /**
     * If an invalid or illegal string is specified.
     * @since DOM Level 2
     */
    public static final short SYNTAX_ERR                = 12;
    /**
     * If an attempt is made to modify the type of the underlying object.
     * @since DOM Level 2
     */
    public static final short INVALID_MODIFICATION_ERR  = 13;
    /**
     * If an attempt is made to create or change an object in a way which is
     * incorrect with regard to namespaces.
     * @since DOM Level 2
     */
    public static final short NAMESPACE_ERR             = 14;
    /**
     * If a parameter or an operation is not supported by the underlying
     * object.
     * @since DOM Level 2
     */
```

```
    public static final short INVALID_ACCESS_ERR        = 15;
    /**
     * If a call to a method such as <code>insertBefore</code> or
     * <code>removeChild</code> would make the <code>Node</code> invalid
     * with respect to "partial validity", this exception would be raised
     * and the operation would not be done. This code is used in [<a href='http://www.w3.org/TR
         /2004/REC-DOM-Level-3-Val-20040127/'>DOM Level 3 Validation</a>]
     * . Refer to this specification for further information.
     * @since DOM Level 3
     */
    public static final short VALIDATION_ERR            = 16;
    /**
     *  If the type of an object is incompatible with the expected type of the
     * parameter associated to the object.
     * @since DOM Level 3
     */
    public static final short TYPE_MISMATCH_ERR         = 17;

    // Added serialVersionUID to preserve binary compatibility
    static final long serialVersionUID = 6627732366795969916L;
}
```

36.9 DOMImplementation.java

Secuencia 340: org/w3c/dom/DOMImplementation.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>DOMImplementation</code> interface provides a number of methods
 * for performing operations that are independent of any particular instance
 * of the document object model.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface DOMImplementation {
    /**
     * Test if the DOM implementation implements a specific feature and
     * version, as specified in <a href="http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/
         core.html#DOMFeatures">DOM Features</a>.
     * @param feature  The name of the feature to test.
     * @param version  This is the version number of the feature to test.
     * @return <code>true</code> if the feature is implemented in the
     *   specified version, <code>false</code> otherwise.
     */
    public boolean hasFeature(String feature,
                              String version);

    /**
     * Creates an empty <code>DocumentType</code> node. Entity declarations
     * and notations are not made available. Entity reference expansions and
     * default attribute additions do not occur..
     * @param qualifiedName The qualified name of the document type to be
     *   created.
     * @param publicId The external subset public identifier.
     * @param systemId The external subset system identifier.
     * @return A new <code>DocumentType</code> node with
     *   <code>Node.ownerDocument</code> set to <code>null</code>.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified qualified name is not
     *   an XML name according to [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a
         >].
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is
     *   malformed.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature "XML" and the language exposed through the
```

```
     *   Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-
         html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public DocumentType createDocumentType(String qualifiedName,
                                           String publicId,
                                           String systemId)
                                           throws DOMException;

    /**
     * Creates a DOM Document object of the specified type with its document
     * element.
     * <br>Note that based on the <code>DocumentType</code> given to create
     * the document, the implementation may instantiate specialized
     * <code>Document</code> objects that support additional features than
     * the "Core", such as "HTML" [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML
         -20030109'>DOM Level 2 HTML</a>]
     * . On the other hand, setting the <code>DocumentType</code> after the
     * document was created makes this very unlikely to happen.
     * Alternatively, specialized <code>Document</code> creation methods,
     * such as <code>createHTMLDocument</code> [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-
         HTML-20030109'>DOM Level 2 HTML</a>]
     * , can be used to obtain specific types of <code>Document</code>
     * objects.
     * @param namespaceURI The namespace URI of the document element to
     *   create or <code>null</code>.
     * @param qualifiedName The qualified name of the document element to be
     *   created or <code>null</code>.
     * @param doctype The type of document to be created or <code>null</code>.
     *   When <code>doctype</code> is not <code>null</code>, its
     *   <code>Node.ownerDocument</code> attribute is set to the document
     *   being created.
     * @return A new <code>Document</code> object with its document element.
     *   If the <code>NamespaceURI</code>, <code>qualifiedName</code>, and
     *   <code>doctype</code> are <code>null</code>, the returned
     *   <code>Document</code> is empty with no document element.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified qualified name is not
     *   an XML name according to [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a
         >].
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is
     *   malformed, if the <code>qualifiedName</code> has a prefix and the
     *   <code>namespaceURI</code> is <code>null</code>, or if the
     *   <code>qualifiedName</code> is <code>null</code> and the
     *   <code>namespaceURI</code> is different from <code>null</code>, or
     *   if the <code>qualifiedName</code> has a prefix that is "xml" and
     *   the <code>namespaceURI</code> is different from "<a href='http://www.w3.org/XML/1998/
         namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>" [<a href='http://www.w3.org/TR/1999/REC-xml-
         names-19990114/'>XML Namespaces</a>]
     *   , or if the DOM implementation does not support the
     *   <code>"XML"</code> feature but a non-null namespace URI was
     *   provided, since namespaces were defined by XML.
```

```
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>doctype</code> has already
     *   been used with a different document or was created from a different
     *   implementation.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature "XML" and the language exposed through the
     *   Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-
         html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Document createDocument(String namespaceURI,
                                   String qualifiedName,
                                   DocumentType doctype)
                                   throws DOMException;

    /**
     *  This method returns a specialized object which implements the
     * specialized APIs of the specified feature and version, as specified
     * in <a href="http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#DOMFeatures">
         DOM Features</a>. The specialized object may also be obtained by using
     * binding-specific casting methods but is not necessarily expected to,
     * as discussed in . This method also allow the implementation to
     * provide specialized objects which do not support the
     * <code>DOMImplementation</code> interface.
     * @param feature  The name of the feature requested. Note that any plus
     *   sign "+" prepended to the name of the feature will be ignored since
     *   it is not significant in the context of this method.
     * @param version  This is the version number of the feature to test.
     * @return  Returns an object which implements the specialized APIs of
     *   the specified feature and version, if any, or <code>null</code> if
     *   there is no object which implements interfaces associated with that
     *   feature. If the <code>DOMObject</code> returned by this method
     *   implements the <code>DOMImplementation</code> interface, it must
     *   delegate to the primary core <code>DOMImplementation</code> and not
     *   return results inconsistent with the primary core
     *   <code>DOMImplementation</code> such as <code>hasFeature</code>,
     *   <code>getFeature</code>, etc.
     * @since DOM Level 3
     */
    public Object getFeature(String feature,
                             String version);

}
```

36.10 DOMImplementationList.java

Secuencia 341: org/w3c/dom/DOMImplementationList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  The <code>DOMImplementationList</code> interface provides the abstraction
 * of an ordered collection of DOM implementations, without defining or
 * constraining how this collection is implemented. The items in the
 * <code>DOMImplementationList</code> are accessible via an integral index,
 * starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMImplementationList {
    /**
     *  Returns the <code>index</code>th item in the collection. If
     * <code>index</code> is greater than or equal to the number of
     * <code>DOMImplementation</code>s in the list, this returns
     * <code>null</code>.
     * @param index Index into the collection.
```

```
     * @return  The <code>DOMImplementation</code> at the <code>index</code>
     *   th position in the <code>DOMImplementationList</code>, or
     *   <code>null</code> if that is not a valid index.
     */
    public DOMImplementation item(int index);

    /**
     *  The number of <code>DOMImplementation</code>s in the list. The range
     * of valid child node indices is 0 to <code>length-1</code> inclusive.
     */
    public int getLength();

}
```

36.11 DOMImplementationSource.java

Secuencia 342: org/w3c/dom/DOMImplementationSource.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
```

```
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * This interface permits a DOM implementer to supply one or more
 * implementations, based upon requested features and versions, as specified
 * in <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#DOMFeatures'>DOM
 * Features</a>. Each implemented <code>DOMImplementationSource</code> object is
 * listed in the binding-specific list of available sources so that its
 * <code>DOMImplementation</code> objects are made available.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMImplementationSource {
    /**
     *  A method to request the first DOM implementation that supports the
     * specified features.
     * @param features  A string that specifies which features and versions
     *   are required. This is a space separated list in which each feature
     *   is specified by its name optionally followed by a space and a
     *   version number.  This method returns the first item of the list
     *   returned by <code>getDOMImplementationList</code>.  As an example,
     *   the string <code>"XML 3.0 Traversal +Events 2.0"</code> will
     *   request a DOM implementation that supports the module "XML" for its
     *   3.0 version, a module that support of the "Traversal" module for
     *   any version, and the module "Events" for its 2.0 version. The
     *   module "Events" must be accessible using the method
     *   <code>Node.getFeature()</code> and
     *   <code>DOMImplementation.getFeature()</code>.
     * @return The first DOM implementation that support the desired
     *   features, or <code>null</code> if this source has none.
     */
    public DOMImplementation getDOMImplementation(String features);

    /**
     * A method to request a list of DOM implementations that support the
     * specified features and versions, as specified in .
     * @param features A string that specifies which features and versions
     *   are required. This is a space separated list in which each feature
     *   is specified by its name optionally followed by a space and a
     *   version number. This is something like: "XML 3.0 Traversal +Events
     *   2.0"
     * @return A list of DOM implementations that support the desired
     *   features.
     */
    public DOMImplementationList getDOMImplementationList(String features);

}
```

36.12 DOMLocator.java

Secuencia 343: org/w3c/dom/DOMLocator.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * <code>DOMLocator</code> is an interface that describes a location (e.g.
 * where an error occurred).
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
```

```
public interface DOMLocator {
    /**
     * The line number this locator is pointing to, or <code>-1</code> if
     * there is no column number available.
     */
    public int getLineNumber();

    /**
     * The column number this locator is pointing to, or <code>-1</code> if
     * there is no column number available.
     */
    public int getColumnNumber();

    /**
     * The byte offset into the input source this locator is pointing to or
     * <code>-1</code> if there is no byte offset available.
     */
    public int getByteOffset();

    /**
     * The UTF-16, as defined in [Unicode] and Amendment 1 of [ISO/IEC 10646], offset into the
         input source this locator is pointing to or
     * <code>-1</code> if there is no UTF-16 offset available.
     */
    public int getUtf16Offset();

    /**
     * The node this locator is pointing to, or <code>null</code> if no node
     * is available.
     */
    public Node getRelatedNode();

    /**
     * The URI this locator is pointing to, or <code>null</code> if no URI is
     * available.
     */
    public String getUri();

}
```

36.13 DOMStringList.java

Secuencia 344: org/w3c/dom/DOMStringList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  The <code>DOMStringList</code> interface provides the abstraction of an
 * ordered collection of <code>DOMString</code> values, without defining or
 * constraining how this collection is implemented. The items in the
 * <code>DOMStringList</code> are accessible via an integral index, starting
 * from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface DOMStringList {
    /**
     *  Returns the <code>index</code>th item in the collection. If
     * <code>index</code> is greater than or equal to the number of
     * <code>DOMString</code>s in the list, this returns <code>null</code>.
     * @param index Index into the collection.
     * @return  The <code>DOMString</code> at the <code>index</code>th
     *   position in the <code>DOMStringList</code>, or <code>null</code> if
     *   that is not a valid index.
     */
```

```
    public String item(int index);

    /**
     * The number of <code>DOMString</code>s in the list. The range of valid
     * child node indices is 0 to <code>length-1</code> inclusive.
     */
    public int getLength();

    /**
     *  Test if a string is part of this <code>DOMStringList</code>.
     * @param str  The string to look for.
     * @return  <code>true</code> if the string has been found,
     *   <code>false</code> otherwise.
     */
    public boolean contains(String str);

}
```

36.14 Document.java

Secuencia 345: org/w3c/dom/Document.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
```

```
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>Document</code> interface represents the entire HTML or XML
 * document. Conceptually, it is the root of the document tree, and provides
 * the primary access to the document's data.
 * <p>Since elements, text nodes, comments, processing instructions, etc.
 * cannot exist outside the context of a <code>Document</code>, the
 * <code>Document</code> interface also contains the factory methods needed
 * to create these objects. The <code>Node</code> objects created have a
 * <code>ownerDocument</code> attribute which associates them with the
 * <code>Document</code> within whose context they were created.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Document extends Node {
    /**
     * The Document Type Declaration (see <code>DocumentType</code>)
     * associated with this document. For XML documents without a document
     * type declaration this returns <code>null</code>. For HTML documents,
     * a <code>DocumentType</code> object may be returned, independently of
     * the presence or absence of document type declaration in the HTML
     * document.
     * <br>This provides direct access to the <code>DocumentType</code> node,
     * child node of this <code>Document</code>. This node can be set at
     * document creation time and later changed through the use of child
     * nodes manipulation methods, such as <code>Node.insertBefore</code>,
     * or <code>Node.replaceChild</code>. Note, however, that while some
     * implementations may instantiate different types of
     * <code>Document</code> objects supporting additional features than the
     * "Core", such as "HTML" [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109'>
         DOM Level 2 HTML</a>]
     * , based on the <code>DocumentType</code> specified at creation time,
     * changing it afterwards is very unlikely to result in a change of the
     * features supported.
     *
     * @since DOM Level 3
     */
    public DocumentType getDoctype();

    /**
     * The <code>DOMImplementation</code> object that handles this document. A
     * DOM application may use objects from multiple implementations.
     */
```

```
    public DOMImplementation getImplementation();

    /**
     * This is a convenience attribute that allows direct access to the child
     * node that is the document element of the document.
     */
    public Element getDocumentElement();

    /**
     * Creates an element of the type specified. Note that the instance
     * returned implements the <code>Element</code> interface, so attributes
     * can be specified directly on the returned object.
     * <br>In addition, if there are known attributes with default values,
     * <code>Attr</code> nodes representing them are automatically created
     * and attached to the element.
     * <br>To create an element with a qualified name and namespace URI, use
     * the <code>createElementNS</code> method.
     * @param tagName The name of the element type to instantiate. For XML,
     *   this is case-sensitive, otherwise it depends on the
     *   case-sensitivity of the markup language in use. In that case, the
     *   name is mapped to the canonical form of that markup by the DOM
     *   implementation.
     * @return A new <code>Element</code> object with the
     *   <code>nodeName</code> attribute set to <code>tagName</code>, and
     *   <code>localName</code>, <code>prefix</code>, and
     *   <code>namespaceURI</code> set to <code>null</code>.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified name is not an XML
     *   name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     */
    public Element createElement(String tagName)
                                 throws DOMException;

    /**
     * Creates an empty <code>DocumentFragment</code> object.
     * @return A new <code>DocumentFragment</code>.
     */
    public DocumentFragment createDocumentFragment();

    /**
     * Creates a <code>Text</code> node given the specified string.
     * @param data The data for the node.
     * @return The new <code>Text</code> object.
     */
    public Text createTextNode(String data);

    /**
     * Creates a <code>Comment</code> node given the specified string.
     * @param data The data for the node.
     * @return The new <code>Comment</code> object.
     */
    public Comment createComment(String data);
```

```java

    /**
     * Creates a <code>CDATASection</code> node whose value is the specified
     * string.
     * @param data The data for the <code>CDATASection</code> contents.
     * @return The new <code>CDATASection</code> object.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if this document is an HTML document.
     */
    public CDATASection createCDATASection(String data)
                                           throws DOMException;

    /**
     * Creates a <code>ProcessingInstruction</code> node given the specified
     * name and data strings.
     * @param target The target part of the processing instruction.Unlike
     *   <code>Document.createElementNS</code> or
     *   <code>Document.createAttributeNS</code>, no namespace well-formed
     *   checking is done on the target name. Applications should invoke
     *   <code>Document.normalizeDocument()</code> with the parameter "
     *   namespaces" set to <code>true</code> in order to ensure that the
     *   target name is namespace well-formed.
     * @param data The data for the node.
     * @return The new <code>ProcessingInstruction</code> object.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified target is not an XML
     *   name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     *   <br>NOT_SUPPORTED_ERR: Raised if this document is an HTML document.
     */
    public ProcessingInstruction createProcessingInstruction(String target,
                                                             String data)
                                                             throws DOMException;

    /**
     * Creates an <code>Attr</code> of the given name. Note that the
     * <code>Attr</code> instance can then be set on an <code>Element</code>
     * using the <code>setAttributeNode</code> method.
     * <br>To create an attribute with a qualified name and namespace URI, use
     * the <code>createAttributeNS</code> method.
     * @param name The name of the attribute.
     * @return A new <code>Attr</code> object with the <code>nodeName</code>
     *   attribute set to <code>name</code>, and <code>localName</code>,
     *   <code>prefix</code>, and <code>namespaceURI</code> set to
     *   <code>null</code>. The value of the attribute is the empty string.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified name is not an XML
     *   name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     */
    public Attr createAttribute(String name)
                                throws DOMException;

```

```
    /**
     * Creates an <code>EntityReference</code> object. In addition, if the
     * referenced entity is known, the child list of the
     * <code>EntityReference</code> node is made the same as that of the
     * corresponding <code>Entity</code> node.
     * <p ><b>Note:</b> If any descendant of the <code>Entity</code> node has
     * an unbound namespace prefix, the corresponding descendant of the
     * created <code>EntityReference</code> node is also unbound; (its
     * <code>namespaceURI</code> is <code>null</code>). The DOM Level 2 and
     * 3 do not support any mechanism to resolve namespace prefixes in this
     * case.
     * @param name The name of the entity to reference.Unlike
     *   <code>Document.createElementNS</code> or
     *   <code>Document.createAttributeNS</code>, no namespace well-formed
     *   checking is done on the entity name. Applications should invoke
     *   <code>Document.normalizeDocument()</code> with the parameter "
     *   namespaces" set to <code>true</code> in order to ensure that the
     *   entity name is namespace well-formed.
     * @return The new <code>EntityReference</code> object.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified name is not an XML
     *   name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     *   <br>NOT_SUPPORTED_ERR: Raised if this document is an HTML document.
     */
    public EntityReference createEntityReference(String name)
                                                 throws DOMException;

    /**
     * Returns a <code>NodeList</code> of all the <code>Elements</code> in
     * document order with a given tag name and are contained in the
     * document.
     * @param tagname  The name of the tag to match on. The special value "*"
     *   matches all tags. For XML, the <code>tagname</code> parameter is
     *   case-sensitive, otherwise it depends on the case-sensitivity of the
     *   markup language in use.
     * @return A new <code>NodeList</code> object containing all the matched
     *   <code>Elements</code>.
     */
    public NodeList getElementsByTagName(String tagname);

    /**
     * Imports a node from another document to this document, without altering
     * or removing the source node from the original document; this method
     * creates a new copy of the source node. The returned node has no
     * parent; (<code>parentNode</code> is <code>null</code>).
     * <br>For all nodes, importing a node creates a node object owned by the
     * importing document, with attribute values identical to the source
     * node's <code>nodeName</code> and <code>nodeType</code>, plus the
     * attributes related to namespaces (<code>prefix</code>,
     * <code>localName</code>, and <code>namespaceURI</code>). As in the
     * <code>cloneNode</code> operation, the source node is not altered.
     * User data associated to the imported node is not carried over.
```

```
     * However, if any <code>UserDataHandlers</code> has been specified
     * along with the associated data these handlers will be called with the
     * appropriate parameters before this method returns.
     * <br>Additional information is copied as appropriate to the
     * <code>nodeType</code>, attempting to mirror the behavior expected if
     * a fragment of XML or HTML source was copied from one document to
     * another, recognizing that the two documents may have different DTDs
     * in the XML case. The following list describes the specifics for each
     * type of node.
     * <dl>
     * <dt>ATTRIBUTE_NODE</dt>
     * <dd>The <code>ownerElement</code> attribute
     * is set to <code>null</code> and the <code>specified</code> flag is
     * set to <code>true</code> on the generated <code>Attr</code>. The
     * descendants of the source <code>Attr</code> are recursively imported
     * and the resulting nodes reassembled to form the corresponding subtree.
     * Note that the <code>deep</code> parameter has no effect on
     * <code>Attr</code> nodes; they always carry their children with them
     * when imported.</dd>
     * <dt>DOCUMENT_FRAGMENT_NODE</dt>
     * <dd>If the <code>deep</code> option
     * was set to <code>true</code>, the descendants of the source
     * <code>DocumentFragment</code> are recursively imported and the
     * resulting nodes reassembled under the imported
     * <code>DocumentFragment</code> to form the corresponding subtree.
     * Otherwise, this simply generates an empty
     * <code>DocumentFragment</code>.</dd>
     * <dt>DOCUMENT_NODE</dt>
     * <dd><code>Document</code>
     * nodes cannot be imported.</dd>
     * <dt>DOCUMENT_TYPE_NODE</dt>
     * <dd><code>DocumentType</code>
     * nodes cannot be imported.</dd>
     * <dt>ELEMENT_NODE</dt>
     * <dd><em>Specified</em> attribute nodes of the source element are imported, and the generated
     * <code>Attr</code> nodes are attached to the generated
     * <code>Element</code>. Default attributes are <em>not</em> copied, though if the document
       being imported into defines default
     * attributes for this element name, those are assigned. If the
     * <code>importNode</code> <code>deep</code> parameter was set to
     * <code>true</code>, the descendants of the source element are
     * recursively imported and the resulting nodes reassembled to form the
     * corresponding subtree.</dd>
     * <dt>ENTITY_NODE</dt>
     * <dd><code>Entity</code> nodes can be
     * imported, however in the current release of the DOM the
     * <code>DocumentType</code> is readonly. Ability to add these imported
     * nodes to a <code>DocumentType</code> will be considered for addition
     * to a future release of the DOM.On import, the <code>publicId</code>,
     * <code>systemId</code>, and <code>notationName</code> attributes are
     * copied. If a <code>deep</code> import is requested, the descendants
     * of the the source <code>Entity</code> are recursively imported and
     * the resulting nodes reassembled to form the corresponding subtree.</dd>
```

```
     * <dt>
     * ENTITY_REFERENCE_NODE</dt>
     * <dd>Only the <code>EntityReference</code> itself is
     * copied, even if a <code>deep</code> import is requested, since the
     * source and destination documents might have defined the entity
     * differently. If the document being imported into provides a
     * definition for this entity name, its value is assigned.</dd>
     * <dt>NOTATION_NODE</dt>
     * <dd>
     * <code>Notation</code> nodes can be imported, however in the current
     * release of the DOM the <code>DocumentType</code> is readonly. Ability
     * to add these imported nodes to a <code>DocumentType</code> will be
     * considered for addition to a future release of the DOM.On import, the
     * <code>publicId</code> and <code>systemId</code> attributes are copied.
     * Note that the <code>deep</code> parameter has no effect on this type
     * of nodes since they cannot have any children.</dd>
     * <dt>
     * PROCESSING_INSTRUCTION_NODE</dt>
     * <dd>The imported node copies its
     * <code>target</code> and <code>data</code> values from those of the
     * source node.Note that the <code>deep</code> parameter has no effect
     * on this type of nodes since they cannot have any children.</dd>
     * <dt>TEXT_NODE,
     * CDATA_SECTION_NODE, COMMENT_NODE</dt>
     * <dd>These three types of nodes inheriting
     * from <code>CharacterData</code> copy their <code>data</code> and
     * <code>length</code> attributes from those of the source node.Note
     * that the <code>deep</code> parameter has no effect on these types of
     * nodes since they cannot have any children.</dd>
     * </dl>
     * @param importedNode The node to import.
     * @param deep If <code>true</code>, recursively import the subtree under
     *   the specified node; if <code>false</code>, import only the node
     *   itself, as explained above. This has no effect on nodes that cannot
     *   have any children, and on <code>Attr</code>, and
     *   <code>EntityReference</code> nodes.
     * @return The imported node that belongs to this <code>Document</code>.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if the type of node being imported is not
     *   supported.
     *   <br>INVALID_CHARACTER_ERR: Raised if one of the imported names is not
     *   an XML name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute. This may happen when
     *   importing an XML 1.1 [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>] element
     *   into an XML 1.0 document, for instance.
     * @since DOM Level 2
     */
    public Node importNode(Node importedNode,
                           boolean deep)
                           throws DOMException;

    /**
```

```
     * Creates an element of the given qualified name and namespace URI.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * namespaceURI parameter for methods if they wish to have no namespace.
     * @param namespaceURI The namespace URI of the element to create.
     * @param qualifiedName The qualified name of the element type to
     *   instantiate.
     * @return A new <code>Element</code> object with the following
     *   attributes:
     * <table border='1' cellpadding='3'>
     * <tr>
     * <th>Attribute</th>
     * <th>Value</th>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.nodeName</code></td>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>qualifiedName</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.namespaceURI</code></td>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>namespaceURI</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.prefix</code></td>
     * <td valign='top' rowspan='1' colspan='1'>prefix, extracted
     *   from <code>qualifiedName</code>, or <code>null</code> if there is
     *   no prefix</td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.localName</code></td>
     * <td valign='top' rowspan='1' colspan='1'>local name, extracted from
     *   <code>qualifiedName</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Element.tagName</code></td>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>qualifiedName</code></td>
     * </tr>
     * </table>
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified
     *   <code>qualifiedName</code> is not an XML name according to the XML
     *   version in use specified in the <code>Document.xmlVersion</code>
     *   attribute.
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is a
     *   malformed qualified name, if the <code>qualifiedName</code> has a
     *   prefix and the <code>namespaceURI</code> is <code>null</code>, or
     *   if the <code>qualifiedName</code> has a prefix that is "xml" and
     *   the <code>namespaceURI</code> is different from "<a href='http://www.w3.org/XML/1998/
         namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>" [<a href='http://www.w3.org/TR/1999/REC-xml-
```

```
        names-19990114/'>XML Namespaces</a>]
     *   , or if the <code>qualifiedName</code> or its prefix is "xmlns" and
     *   the <code>namespaceURI</code> is different from "<a href='http://www.w3.org/2000/xmlns/'>
         http://www.w3.org/2000/xmlns/</a>", or if the <code>namespaceURI</code> is "<a href='http
         ://www.w3.org/2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>" and neither the <code>
         qualifiedName</code> nor its prefix is "xmlns".
     *   <br>NOT_SUPPORTED_ERR: Always thrown if the current document does not
     *   support the <code>"XML"</code> feature, since namespaces were
     *   defined by XML.
     * @since DOM Level 2
     */
    public Element createElementNS(String namespaceURI,
                                   String qualifiedName)
                                   throws DOMException;

    /**
     * Creates an attribute of the given qualified name and namespace URI.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to create.
     * @param qualifiedName The qualified name of the attribute to
     *   instantiate.
     * @return A new <code>Attr</code> object with the following attributes:
     * <table border='1' cellpadding='3'>
     * <tr>
     * <th>
     *   Attribute</th>
     * <th>Value</th>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.nodeName</code></td>
     * <td valign='top' rowspan='1' colspan='1'>qualifiedName</td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>Node.namespaceURI</code></td>
     * <td valign='top' rowspan='1' colspan='1'><code>namespaceURI</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>Node.prefix</code></td>
     * <td valign='top' rowspan='1' colspan='1'>prefix, extracted from
     *   <code>qualifiedName</code>, or <code>null</code> if there is no
     *   prefix</td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.localName</code></td>
     * <td valign='top' rowspan='1' colspan='1'>local name, extracted from
     *   <code>qualifiedName</code></td>
     * </tr>
     * <tr>
```

```
     * <td valign='top' rowspan='1' colspan='1'><code>Attr.name</code></td>
     * <td valign='top' rowspan='1' colspan='1'>
     *   <code>qualifiedName</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'><code>Node.nodeValue</code></td>
     * <td valign='top' rowspan='1' colspan='1'>the empty
     *   string</td>
     * </tr>
     * </table>
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified
     *   <code>qualifiedName</code> is not an XML name according to the XML
     *   version in use specified in the <code>Document.xmlVersion</code>
     *   attribute.
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is a
     *   malformed qualified name, if the <code>qualifiedName</code> has a
     *   prefix and the <code>namespaceURI</code> is <code>null</code>, if
     *   the <code>qualifiedName</code> has a prefix that is "xml" and the
     *   <code>namespaceURI</code> is different from "<a href='http://www.w3.org/XML/1998/namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>", if the <code>qualifiedName</code> or its prefix is "xmlns" and the
     *   <code>namespaceURI</code> is different from "<a href='http://www.w3.org/2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>", or if the <code>namespaceURI</code> is "<a href='http://www.w3.org/2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>" and neither the <code>qualifiedName</code> nor its prefix is "xmlns".
     *   <br>NOT_SUPPORTED_ERR: Always thrown if the current document does not
     *   support the <code>"XML"</code> feature, since namespaces were
     *   defined by XML.
     * @since DOM Level 2
     */
    public Attr createAttributeNS(String namespaceURI,
                                  String qualifiedName)
                                  throws DOMException;

    /**
     * Returns a <code>NodeList</code> of all the <code>Elements</code> with a
     * given local name and namespace URI in document order.
     * @param namespaceURI The namespace URI of the elements to match on. The
     *   special value <code>"*"</code> matches all namespaces.
     * @param localName The local name of the elements to match on. The
     *   special value "*" matches all local names.
     * @return A new <code>NodeList</code> object containing all the matched
     *   <code>Elements</code>.
     * @since DOM Level 2
     */
    public NodeList getElementsByTagNameNS(String namespaceURI,
                                           String localName);

    /**
     * Returns the <code>Element</code> that has an ID attribute with the
     * given value. If no such element exists, this returns <code>null</code>
```

```
     * . If more than one element has an ID attribute with that value, what
     * is returned is undefined.
     * <br> The DOM implementation is expected to use the attribute
     * <code>Attr.isId</code> to determine if an attribute is of type ID.
     * <p ><b>Note:</b> Attributes with the name "ID" or "id" are not of type
     * ID unless so defined.
     * @param elementId The unique <code>id</code> value for an element.
     * @return The matching element or <code>null</code> if there is none.
     * @since DOM Level 2
     */
    public Element getElementById(String elementId);

    /**
     * An attribute specifying the encoding used for this document at the time
     * of the parsing. This is <code>null</code> when it is not known, such
     * as when the <code>Document</code> was created in memory.
     * @since DOM Level 3
     */
    public String getInputEncoding();

    /**
     * An attribute specifying, as part of the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#
         NT-XMLDecl'>XML declaration</a>, the encoding of this document. This is <code>null</code>
         when
     * unspecified or when it is not known, such as when the
     * <code>Document</code> was created in memory.
     * @since DOM Level 3
     */
    public String getXmlEncoding();

    /**
     * An attribute specifying, as part of the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#
         NT-XMLDecl'>XML declaration</a>, whether this document is standalone. This is <code>false
         </code> when
     * unspecified.
     * <p ><b>Note:</b>  No verification is done on the value when setting
     * this attribute. Applications should use
     * <code>Document.normalizeDocument()</code> with the "validate"
     * parameter to verify if the value matches the <a href='http://www.w3.org/TR/2004/REC-xml
         -20040204#sec-rmd'>validity
     * constraint for standalone document declaration</a> as defined in [<a href='http://www.w3.org
         /TR/2004/REC-xml-20040204'>XML 1.0</a>].
     * @since DOM Level 3
     */
    public boolean getXmlStandalone();
    /**
     * An attribute specifying, as part of the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#
         NT-XMLDecl'>XML declaration</a>, whether this document is standalone. This is <code>false
         </code> when
     * unspecified.
     * <p ><b>Note:</b>  No verification is done on the value when setting
     * this attribute. Applications should use
     * <code>Document.normalizeDocument()</code> with the "validate"
```

```
     * parameter to verify if the value matches the <a href='http://www.w3.org/TR/2004/REC-xml
         -20040204#sec-rmd'>validity
     * constraint for standalone document declaration</a> as defined in [<a href='http://www.w3.org
         /TR/2004/REC-xml-20040204'>XML 1.0</a>].
     * @exception DOMException
     *    NOT_SUPPORTED_ERR: Raised if this document does not support the
     *   "XML" feature.
     * @since DOM Level 3
     */
    public void setXmlStandalone(boolean xmlStandalone)
                                 throws DOMException;

    /**
     *  An attribute specifying, as part of the <a href='http://www.w3.org/TR/2004/REC-xml
         -20040204#NT-XMLDecl'>XML declaration</a>, the version number of this document. If there
         is no declaration and if
     * this document supports the "XML" feature, the value is
     * <code>"1.0"</code>. If this document does not support the "XML"
     * feature, the value is always <code>null</code>. Changing this
     * attribute will affect methods that check for invalid characters in
     * XML names. Application should invoke
     * <code>Document.normalizeDocument()</code> in order to check for
     * invalid characters in the <code>Node</code>s that are already part of
     * this <code>Document</code>.
     * <br> DOM applications may use the
     * <code>DOMImplementation.hasFeature(feature, version)</code> method
     * with parameter values "XMLVersion" and "1.0" (respectively) to
     * determine if an implementation supports [<a href='http://www.w3.org/TR/2004/REC-xml
         -20040204'>XML 1.0</a>]. DOM
     * applications may use the same method with parameter values
     * "XMLVersion" and "1.1" (respectively) to determine if an
     * implementation supports [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a
         >]. In both
     * cases, in order to support XML, an implementation must also support
     * the "XML" feature defined in this specification. <code>Document</code>
     *  objects supporting a version of the "XMLVersion" feature must not
     * raise a <code>NOT_SUPPORTED_ERR</code> exception for the same version
     * number when using <code>Document.xmlVersion</code>.
     * @since DOM Level 3
     */
    public String getXmlVersion();
    /**
     *  An attribute specifying, as part of the <a href='http://www.w3.org/TR/2004/REC-xml
         -20040204#NT-XMLDecl'>XML declaration</a>, the version number of this document. If there
         is no declaration and if
     * this document supports the "XML" feature, the value is
     * <code>"1.0"</code>. If this document does not support the "XML"
     * feature, the value is always <code>null</code>. Changing this
     * attribute will affect methods that check for invalid characters in
     * XML names. Application should invoke
     * <code>Document.normalizeDocument()</code> in order to check for
     * invalid characters in the <code>Node</code>s that are already part of
     * this <code>Document</code>.
```

```
     * <br> DOM applications may use the
     * <code>DOMImplementation.hasFeature(feature, version)</code> method
     * with parameter values "XMLVersion" and "1.0" (respectively) to
     * determine if an implementation supports [<a href='http://www.w3.org/TR/2004/REC-xml
         -20040204'>XML 1.0</a>]. DOM
     * applications may use the same method with parameter values
     * "XMLVersion" and "1.1" (respectively) to determine if an
     * implementation supports [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a
         >]. In both
     * cases, in order to support XML, an implementation must also support
     * the "XML" feature defined in this specification. <code>Document</code>
     *  objects supporting a version of the "XMLVersion" feature must not
     * raise a <code>NOT_SUPPORTED_ERR</code> exception for the same version
     * number when using <code>Document.xmlVersion</code>.
     * @exception DOMException
     *    NOT_SUPPORTED_ERR: Raised if the version is set to a value that is
     *   not supported by this <code>Document</code> or if this document
     *   does not support the "XML" feature.
     * @since DOM Level 3
     */
    public void setXmlVersion(String xmlVersion)
                                throws DOMException;

    /**
     * An attribute specifying whether error checking is enforced or not. When
     * set to <code>false</code>, the implementation is free to not test
     * every possible error case normally defined on DOM operations, and not
     * raise any <code>DOMException</code> on DOM operations or report
     * errors while using <code>Document.normalizeDocument()</code>. In case
     * of error, the behavior is undefined. This attribute is
     * <code>true</code> by default.
     * @since DOM Level 3
     */
    public boolean getStrictErrorChecking();
    /**
     * An attribute specifying whether error checking is enforced or not. When
     * set to <code>false</code>, the implementation is free to not test
     * every possible error case normally defined on DOM operations, and not
     * raise any <code>DOMException</code> on DOM operations or report
     * errors while using <code>Document.normalizeDocument()</code>. In case
     * of error, the behavior is undefined. This attribute is
     * <code>true</code> by default.
     * @since DOM Level 3
     */
    public void setStrictErrorChecking(boolean strictErrorChecking);

    /**
     *  The location of the document or <code>null</code> if undefined or if
     * the <code>Document</code> was created using
     * <code>DOMImplementation.createDocument</code>. No lexical checking is
     * performed when setting this attribute; this could result in a
     * <code>null</code> value returned when using <code>Node.baseURI</code>
     * .
```

```
     * <br> Beware that when the <code>Document</code> supports the feature
     * "HTML" [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109'>DOM Level 2 HTML</
         a>]
     * , the href attribute of the HTML BASE element takes precedence over
     * this attribute when computing <code>Node.baseURI</code>.
     * @since DOM Level 3
     */
    public String getDocumentURI();
    /**
     *  The location of the document or <code>null</code> if undefined or if
     * the <code>Document</code> was created using
     * <code>DOMImplementation.createDocument</code>. No lexical checking is
     * performed when setting this attribute; this could result in a
     * <code>null</code> value returned when using <code>Node.baseURI</code>
     * .
     * <br> Beware that when the <code>Document</code> supports the feature
     * "HTML" [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109'>DOM Level 2 HTML</
         a>]
     * , the href attribute of the HTML BASE element takes precedence over
     * this attribute when computing <code>Node.baseURI</code>.
     * @since DOM Level 3
     */
    public void setDocumentURI(String documentURI);

    /**
     *  Attempts to adopt a node from another document to this document. If
     * supported, it changes the <code>ownerDocument</code> of the source
     * node, its children, as well as the attached attribute nodes if there
     * are any. If the source node has a parent it is first removed from the
     * child list of its parent. This effectively allows moving a subtree
     * from one document to another (unlike <code>importNode()</code> which
     * create a copy of the source node instead of moving it). When it
     * fails, applications should use <code>Document.importNode()</code>
     * instead. Note that if the adopted node is already part of this
     * document (i.e. the source and target document are the same), this
     * method still has the effect of removing the source node from the
     * child list of its parent, if any. The following list describes the
     * specifics for each type of node.
     * <dl>
     * <dt>ATTRIBUTE_NODE</dt>
     * <dd>The
     * <code>ownerElement</code> attribute is set to <code>null</code> and
     * the <code>specified</code> flag is set to <code>true</code> on the
     * adopted <code>Attr</code>. The descendants of the source
     * <code>Attr</code> are recursively adopted.</dd>
     * <dt>DOCUMENT_FRAGMENT_NODE</dt>
     * <dd>The
     * descendants of the source node are recursively adopted.</dd>
     * <dt>DOCUMENT_NODE</dt>
     * <dd>
     * <code>Document</code> nodes cannot be adopted.</dd>
     * <dt>DOCUMENT_TYPE_NODE</dt>
     * <dd>
```

```
     * <code>DocumentType</code> nodes cannot be adopted.</dd>
     * <dt>ELEMENT_NODE</dt>
     * <dd><em>Specified</em> attribute nodes of the source element are adopted. Default attributes
     * are discarded, though if the document being adopted into defines
     * default attributes for this element name, those are assigned. The
     * descendants of the source element are recursively adopted.</dd>
     * <dt>ENTITY_NODE</dt>
     * <dd>
     * <code>Entity</code> nodes cannot be adopted.</dd>
     * <dt>ENTITY_REFERENCE_NODE</dt>
     * <dd>Only
     * the <code>EntityReference</code> node itself is adopted, the
     * descendants are discarded, since the source and destination documents
     * might have defined the entity differently. If the document being
     * imported into provides a definition for this entity name, its value
     * is assigned.</dd>
     * <dt>NOTATION_NODE</dt>
     * <dd><code>Notation</code> nodes cannot be
     * adopted.</dd>
     * <dt>PROCESSING_INSTRUCTION_NODE, TEXT_NODE, CDATA_SECTION_NODE,
     * COMMENT_NODE</dt>
     * <dd>These nodes can all be adopted. No specifics.</dd>
     * </dl>
     * <p ><b>Note:</b>  Since it does not create new nodes unlike the
     * <code>Document.importNode()</code> method, this method does not raise
     * an <code>INVALID_CHARACTER_ERR</code> exception, and applications
     * should use the <code>Document.normalizeDocument()</code> method to
     * check if an imported name is not an XML name according to the XML
     * version in use.
     * @param source The node to move into this document.
     * @return The adopted node, or <code>null</code> if this operation
     *   fails, such as when the source node comes from a different
     *   implementation.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if the source node is of type
     *   <code>DOCUMENT</code>, <code>DOCUMENT_TYPE</code>.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised when the source node is
     *   readonly.
     * @since DOM Level 3
     */
    public Node adoptNode(Node source)
                          throws DOMException;

    /**
     *  The configuration used when <code>Document.normalizeDocument()</code>
     * is invoked.
     * @since DOM Level 3
     */
    public DOMConfiguration getDomConfig();

    /**
     *  This method acts as if the document was going through a save and load
     * cycle, putting the document in a "normal" form. As a consequence,
```

```
     * this method updates the replacement tree of
     * <code>EntityReference</code> nodes and normalizes <code>Text</code>
     * nodes, as defined in the method <code>Node.normalize()</code>.
     * <br> Otherwise, the actual result depends on the features being set on
     * the <code>Document.domConfig</code> object and governing what
     * operations actually take place. Noticeably this method could also
     * make the document namespace well-formed according to the algorithm
     * described in , check the character normalization, remove the
     * <code>CDATASection</code> nodes, etc. See
     * <code>DOMConfiguration</code> for details.
     * <pre>// Keep in the document
     * the information defined // in the XML Information Set (Java example)
     * DOMConfiguration docConfig = myDocument.getDomConfig();
     * docConfig.setParameter("infoset", Boolean.TRUE);
     * myDocument.normalizeDocument();</pre>
     *
     * <br>Mutation events, when supported, are generated to reflect the
     * changes occurring on the document.
     * <br> If errors occur during the invocation of this method, such as an
     * attempt to update a read-only node or a <code>Node.nodeName</code>
     * contains an invalid character according to the XML version in use,
     * errors or warnings (<code>DOMError.SEVERITY_ERROR</code> or
     * <code>DOMError.SEVERITY_WARNING</code>) will be reported using the
     * <code>DOMErrorHandler</code> object associated with the "error-handler
     * " parameter. Note this method might also report fatal errors (
     * <code>DOMError.SEVERITY_FATAL_ERROR</code>) if an implementation
     * cannot recover from an error.
     * @since DOM Level 3
     */
    public void normalizeDocument();

    /**
     * Rename an existing node of type <code>ELEMENT_NODE</code> or
     * <code>ATTRIBUTE_NODE</code>.
     * <br>When possible this simply changes the name of the given node,
     * otherwise this creates a new node with the specified name and
     * replaces the existing node with the new node as described below.
     * <br>If simply changing the name of the given node is not possible, the
     * following operations are performed: a new node is created, any
     * registered event listener is registered on the new node, any user
     * data attached to the old node is removed from that node, the old node
     * is removed from its parent if it has one, the children are moved to
     * the new node, if the renamed node is an <code>Element</code> its
     * attributes are moved to the new node, the new node is inserted at the
     * position the old node used to have in its parent's child nodes list
     * if it has one, the user data that was attached to the old node is
     * attached to the new node.
     * <br>When the node being renamed is an <code>Element</code> only the
     * specified attributes are moved, default attributes originated from
     * the DTD are updated according to the new element name. In addition,
     * the implementation may update default attributes from other schemas.
     * Applications should use <code>Document.normalizeDocument()</code> to
     * guarantee these attributes are up-to-date.
```

```
     * <br>When the node being renamed is an <code>Attr</code> that is
     * attached to an <code>Element</code>, the node is first removed from
     * the <code>Element</code> attributes map. Then, once renamed, either
     * by modifying the existing node or creating a new one as described
     * above, it is put back.
     * <br>In addition,
     * <ul>
     * <li> a user data event <code>NODE_RENAMED</code> is fired,
     * </li>
     * <li>
     * when the implementation supports the feature "MutationNameEvents",
     * each mutation operation involved in this method fires the appropriate
     * event, and in the end the event {
     * <code>http://www.w3.org/2001/xml-events</code>,
     * <code>DOMElementNameChanged</code>} or {
     * <code>http://www.w3.org/2001/xml-events</code>,
     * <code>DOMAttributeNameChanged</code>} is fired.
     * </li>
     * </ul>
     * @param n The node to rename.
     * @param namespaceURI The new namespace URI.
     * @param qualifiedName The new qualified name.
     * @return The renamed node. This is either the specified node or the new
     *   node that was created to replace the specified node.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised when the type of the specified node is
     *   neither <code>ELEMENT_NODE</code> nor <code>ATTRIBUTE_NODE</code>,
     *   or if the implementation does not support the renaming of the
     *   document element.
     *   <br>INVALID_CHARACTER_ERR: Raised if the new qualified name is not an
     *   XML name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     *   <br>WRONG_DOCUMENT_ERR: Raised when the specified node was created
     *   from a different document than this document.
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is a
     *   malformed qualified name, if the <code>qualifiedName</code> has a
     *   prefix and the <code>namespaceURI</code> is <code>null</code>, or
     *   if the <code>qualifiedName</code> has a prefix that is "xml" and
     *   the <code>namespaceURI</code> is different from "<a href='http://www.w3.org/XML/1998/
         namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>" [<a href='http://www.w3.org/TR/1999/REC-xml-
         names-19990114/'>XML Namespaces</a>]
     *   . Also raised, when the node being renamed is an attribute, if the
     *   <code>qualifiedName</code>, or its prefix, is "xmlns" and the
     *   <code>namespaceURI</code> is different from "<a href='http://www.w3.org/2000/xmlns/'>http
         ://www.w3.org/2000/xmlns/</a>".
     * @since DOM Level 3
     */
    public Node renameNode(Node n,
                           String namespaceURI,
                           String qualifiedName)
                           throws DOMException;

```

```
}
```

36.15 DocumentFragment.java

Secuencia 346: org/w3c/dom/DocumentFragment.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * <code>DocumentFragment</code> is a "lightweight" or "minimal"
 * <code>Document</code> object. It is very common to want to be able to
 * extract a portion of a document's tree or to create a new fragment of a
 * document. Imagine implementing a user command like cut or rearranging a
```

```
 * document by moving fragments around. It is desirable to have an object
 * which can hold such fragments and it is quite natural to use a Node for
 * this purpose. While it is true that a <code>Document</code> object could
 * fulfill this role, a <code>Document</code> object can potentially be a
 * heavyweight object, depending on the underlying implementation. What is
 * really needed for this is a very lightweight object.
 * <code>DocumentFragment</code> is such an object.
 * <p>Furthermore, various operations -- such as inserting nodes as children
 * of another <code>Node</code> -- may take <code>DocumentFragment</code>
 * objects as arguments; this results in all the child nodes of the
 * <code>DocumentFragment</code> being moved to the child list of this node.
 * <p>The children of a <code>DocumentFragment</code> node are zero or more
 * nodes representing the tops of any sub-trees defining the structure of
 * the document. <code>DocumentFragment</code> nodes do not need to be
 * well-formed XML documents (although they do need to follow the rules
 * imposed upon well-formed XML parsed entities, which can have multiple top
 * nodes). For example, a <code>DocumentFragment</code> might have only one
 * child and that child node could be a <code>Text</code> node. Such a
 * structure model represents neither an HTML document nor a well-formed XML
 * document.
 * <p>When a <code>DocumentFragment</code> is inserted into a
 * <code>Document</code> (or indeed any other <code>Node</code> that may
 * take children) the children of the <code>DocumentFragment</code> and not
 * the <code>DocumentFragment</code> itself are inserted into the
 * <code>Node</code>. This makes the <code>DocumentFragment</code> very
 * useful when the user wishes to create nodes that are siblings; the
 * <code>DocumentFragment</code> acts as the parent of these nodes so that
 * the user can use the standard methods from the <code>Node</code>
 * interface, such as <code>Node.insertBefore</code> and
 * <code>Node.appendChild</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface DocumentFragment extends Node {
}
```

36.16 DocumentType.java

Secuencia 347: org/w3c/dom/DocumentType.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * Each <code>Document</code> has a <code>doctype</code> attribute whose value
 * is either <code>null</code> or a <code>DocumentType</code> object. The
 * <code>DocumentType</code> interface in the DOM Core provides an interface
 * to the list of entities that are defined for the document, and little
 * else because the effect of namespaces and the various XML schema efforts
 * on DTD representation are not clearly understood as of this writing.
 * <p>DOM Level 3 doesn't support editing <code>DocumentType</code> nodes.
 * <code>DocumentType</code> nodes are read-only.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface DocumentType extends Node {
    /**
     * The name of DTD; i.e., the name immediately following the
     * <code>DOCTYPE</code> keyword.
     */
    public String getName();

    /**
     * A <code>NamedNodeMap</code> containing the general entities, both
     * external and internal, declared in the DTD. Parameter entities are
     * not contained. Duplicates are discarded. For example in:
     * <pre>&lt;!DOCTYPE
```

```
     * ex SYSTEM "ex.dtd" [ &lt;!ENTITY foo "foo"&gt; &lt;!ENTITY bar
     * "bar"&gt; &lt;!ENTITY bar "bar2"&gt; &lt;!ENTITY % baz "baz"&gt;
     * ]&gt; &lt;ex/&gt;</pre>
     *  the interface provides access to <code>foo</code>
     * and the first declaration of <code>bar</code> but not the second
     * declaration of <code>bar</code> or <code>baz</code>. Every node in
     * this map also implements the <code>Entity</code> interface.
     * <br>The DOM Level 2 does not support editing entities, therefore
     * <code>entities</code> cannot be altered in any way.
     */
    public NamedNodeMap getEntities();

    /**
     * A <code>NamedNodeMap</code> containing the notations declared in the
     * DTD. Duplicates are discarded. Every node in this map also implements
     * the <code>Notation</code> interface.
     * <br>The DOM Level 2 does not support editing notations, therefore
     * <code>notations</code> cannot be altered in any way.
     */
    public NamedNodeMap getNotations();

    /**
     * The public identifier of the external subset.
     * @since DOM Level 2
     */
    public String getPublicId();

    /**
     * The system identifier of the external subset. This may be an absolute
     * URI or not.
     * @since DOM Level 2
     */
    public String getSystemId();

    /**
     * The internal subset as a string, or <code>null</code> if there is none.
     * This is does not contain the delimiting square brackets.
     * <p ><b>Note:</b> The actual content returned depends on how much
     * information is available to the implementation. This may vary
     * depending on various parameters, including the XML processor used to
     * build the document.
     * @since DOM Level 2
     */
    public String getInternalSubset();

}
```

36.17 Element.java

Secuencia 348: org/w3c/dom/Element.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>Element</code> interface represents an element in an HTML or XML
 * document. Elements may have attributes associated with them; since the
 * <code>Element</code> interface inherits from <code>Node</code>, the
 * generic <code>Node</code> interface attribute <code>attributes</code> may
 * be used to retrieve the set of all attributes for an element. There are
 * methods on the <code>Element</code> interface to retrieve either an
 * <code>Attr</code> object by name or an attribute value by name. In XML,
 * where an attribute value may contain entity references, an
 * <code>Attr</code> object should be retrieved to examine the possibly
 * fairly complex sub-tree representing the attribute value. On the other
 * hand, in HTML, where all attributes have simple string values, methods to
 * directly access an attribute value can safely be used as a convenience.
```

```
 * <p ><b>Note:</b> In DOM Level 2, the method <code>normalize</code> is
 * inherited from the <code>Node</code> interface where it was moved.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Element extends Node {
    /**
     * The name of the element. If <code>Node.localName</code> is different
     * from <code>null</code>, this attribute is a qualified name. For
     * example, in:
     * <pre> &lt;elementExample id="demo"&gt; ...
     * &lt;/elementExample&gt; , </pre>
     *  <code>tagName</code> has the value
     * <code>"elementExample"</code>. Note that this is case-preserving in
     * XML, as are all of the operations of the DOM. The HTML DOM returns
     * the <code>tagName</code> of an HTML element in the canonical
     * uppercase form, regardless of the case in the source HTML document.
     */
    public String getTagName();

    /**
     * Retrieves an attribute value by name.
     * @param name The name of the attribute to retrieve.
     * @return The <code>Attr</code> value as a string, or the empty string
     *   if that attribute does not have a specified or default value.
     */
    public String getAttribute(String name);

    /**
     * Adds a new attribute. If an attribute with that name is already present
     * in the element, its value is changed to be that of the value
     * parameter. This value is a simple string; it is not parsed as it is
     * being set. So any markup (such as syntax to be recognized as an
     * entity reference) is treated as literal text, and needs to be
     * appropriately escaped by the implementation when it is written out.
     * In order to assign an attribute value that contains entity
     * references, the user must create an <code>Attr</code> node plus any
     * <code>Text</code> and <code>EntityReference</code> nodes, build the
     * appropriate subtree, and use <code>setAttributeNode</code> to assign
     * it as the value of an attribute.
     * <br>To set an attribute with a qualified name and namespace URI, use
     * the <code>setAttributeNS</code> method.
     * @param name The name of the attribute to create or alter.
     * @param value Value to set in string form.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified name is not an XML
     *   name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void setAttribute(String name,
                             String value)
                             throws DOMException;
```

```

    /**
     * Removes an attribute by name. If a default value for the removed
     * attribute is defined in the DTD, a new attribute immediately appears
     * with the default value as well as the corresponding namespace URI,
     * local name, and prefix when applicable. The implementation may handle
     * default values from other schemas similarly but applications should
     * use <code>Document.normalizeDocument()</code> to guarantee this
     * information is up-to-date.
     * <br>If no attribute with this name is found, this method has no effect.
     * <br>To remove an attribute by local name and namespace URI, use the
     * <code>removeAttributeNS</code> method.
     * @param name The name of the attribute to remove.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public void removeAttribute(String name)
                                throws DOMException;

    /**
     * Retrieves an attribute node by name.
     * <br>To retrieve an attribute node by qualified name and namespace URI,
     * use the <code>getAttributeNodeNS</code> method.
     * @param name The name (<code>nodeName</code>) of the attribute to
     *   retrieve.
     * @return The <code>Attr</code> node with the specified name (
     *   <code>nodeName</code>) or <code>null</code> if there is no such
     *   attribute.
     */
    public Attr getAttributeNode(String name);

    /**
     * Adds a new attribute node. If an attribute with that name (
     * <code>nodeName</code>) is already present in the element, it is
     * replaced by the new one. Replacing an attribute node by itself has no
     * effect.
     * <br>To add a new attribute node with a qualified name and namespace
     * URI, use the <code>setAttributeNodeNS</code> method.
     * @param newAttr The <code>Attr</code> node to add to the attribute list.
     * @return If the <code>newAttr</code> attribute replaces an existing
     *   attribute, the replaced <code>Attr</code> node is returned,
     *   otherwise <code>null</code> is returned.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: Raised if <code>newAttr</code> was created from a
     *   different document than the one that created the element.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>INUSE_ATTRIBUTE_ERR: Raised if <code>newAttr</code> is already an
     *   attribute of another <code>Element</code> object. The DOM user must
     *   explicitly clone <code>Attr</code> nodes to re-use them in other
     *   elements.
     */
    public Attr setAttributeNode(Attr newAttr)
                                 throws DOMException;
```

```java

    /**
     * Removes the specified attribute node. If a default value for the
     * removed <code>Attr</code> node is defined in the DTD, a new node
     * immediately appears with the default value as well as the
     * corresponding namespace URI, local name, and prefix when applicable.
     * The implementation may handle default values from other schemas
     * similarly but applications should use
     * <code>Document.normalizeDocument()</code> to guarantee this
     * information is up-to-date.
     * @param oldAttr The <code>Attr</code> node to remove from the attribute
     *   list.
     * @return The <code>Attr</code> node that was removed.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if <code>oldAttr</code> is not an attribute
     *   of the element.
     */
    public Attr removeAttributeNode(Attr oldAttr)
                                    throws DOMException;

    /**
     * Returns a <code>NodeList</code> of all descendant <code>Elements</code>
     * with a given tag name, in document order.
     * @param name The name of the tag to match on. The special value "*"
     *   matches all tags.
     * @return A list of matching <code>Element</code> nodes.
     */
    public NodeList getElementsByTagName(String name);

    /**
     * Retrieves an attribute value by local name and namespace URI.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to retrieve.
     * @param localName The local name of the attribute to retrieve.
     * @return The <code>Attr</code> value as a string, or the empty string
     *   if that attribute does not have a specified or default value.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public String getAttributeNS(String namespaceURI,
                                 String localName)
                                 throws DOMException;

    /**
     * Adds a new attribute. If an attribute with the same local name and
```

```
     * namespace URI is already present on the element, its prefix is
     * changed to be the prefix part of the <code>qualifiedName</code>, and
     * its value is changed to be the <code>value</code> parameter. This
     * value is a simple string; it is not parsed as it is being set. So any
     * markup (such as syntax to be recognized as an entity reference) is
     * treated as literal text, and needs to be appropriately escaped by the
     * implementation when it is written out. In order to assign an
     * attribute value that contains entity references, the user must create
     * an <code>Attr</code> node plus any <code>Text</code> and
     * <code>EntityReference</code> nodes, build the appropriate subtree,
     * and use <code>setAttributeNodeNS</code> or
     * <code>setAttributeNode</code> to assign it as the value of an
     * attribute.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to create or
     *   alter.
     * @param qualifiedName The qualified name of the attribute to create or
     *   alter.
     * @param value The value to set in string form.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified qualified name is not
     *   an XML name according to the XML version in use specified in the
     *   <code>Document.xmlVersion</code> attribute.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NAMESPACE_ERR: Raised if the <code>qualifiedName</code> is
     *   malformed per the Namespaces in XML specification, if the
     *   <code>qualifiedName</code> has a prefix and the
     *   <code>namespaceURI</code> is <code>null</code>, if the
     *   <code>qualifiedName</code> has a prefix that is "xml" and the
     *   <code>namespaceURI</code> is different from "<a href='http://www.w3.org/XML/1998/namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>", if the <code>qualifiedName</code> or its prefix is "xmlns" and the
     *   <code>namespaceURI</code> is different from "<a href='http://www.w3.org/2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>", or if the <code>namespaceURI</code> is "<a href='http://www.w3.org/2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>" and neither the <code>qualifiedName</code> nor its prefix is "xmlns".
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public void setAttributeNS(String namespaceURI,
                               String qualifiedName,
                               String value)
                               throws DOMException;

    /**
     * Removes an attribute by local name and namespace URI. If a default
```

```
     * value for the removed attribute is defined in the DTD, a new
     * attribute immediately appears with the default value as well as the
     * corresponding namespace URI, local name, and prefix when applicable.
     * The implementation may handle default values from other schemas
     * similarly but applications should use
     * <code>Document.normalizeDocument()</code> to guarantee this
     * information is up-to-date.
     * <br>If no attribute with this local name and namespace URI is found,
     * this method has no effect.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to remove.
     * @param localName The local name of the attribute to remove.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public void removeAttributeNS(String namespaceURI,
                                  String localName)
                                  throws DOMException;

    /**
     * Retrieves an <code>Attr</code> node by local name and namespace URI.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to retrieve.
     * @param localName The local name of the attribute to retrieve.
     * @return The <code>Attr</code> node with the specified attribute local
     *   name and namespace URI or <code>null</code> if there is no such
     *   attribute.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Attr getAttributeNodeNS(String namespaceURI,
                                   String localName)
                                   throws DOMException;

    /**
     * Adds a new attribute. If an attribute with that local name and that
     * namespace URI is already present in the element, it is replaced by
     * the new one. Replacing an attribute node by itself has no effect.
```

```
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param newAttr The <code>Attr</code> node to add to the attribute list.
     * @return If the <code>newAttr</code> attribute replaces an existing
     *   attribute with the same local name and namespace URI, the replaced
     *   <code>Attr</code> node is returned, otherwise <code>null</code> is
     *   returned.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: Raised if <code>newAttr</code> was created from a
     *   different document than the one that created the element.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>INUSE_ATTRIBUTE_ERR: Raised if <code>newAttr</code> is already an
     *   attribute of another <code>Element</code> object. The DOM user must
     *   explicitly clone <code>Attr</code> nodes to re-use them in other
     *   elements.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Attr setAttributeNodeNS(Attr newAttr)
                                   throws DOMException;

    /**
     * Returns a <code>NodeList</code> of all the descendant
     * <code>Elements</code> with a given local name and namespace URI in
     * document order.
     * @param namespaceURI The namespace URI of the elements to match on. The
     *   special value "*" matches all namespaces.
     * @param localName The local name of the elements to match on. The
     *   special value "*" matches all local names.
     * @return A new <code>NodeList</code> object containing all the matched
     *   <code>Elements</code>.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public NodeList getElementsByTagNameNS(String namespaceURI,
                                           String localName)
                                           throws DOMException;

    /**
     * Returns <code>true</code> when an attribute with a given name is
     * specified on this element or has a default value, <code>false</code>
     * otherwise.
     * @param name The name of the attribute to look for.
     * @return <code>true</code> if an attribute with the given name is
```

```
     *   specified on this element or has a default value, <code>false</code>
     *    otherwise.
     * @since DOM Level 2
     */
    public boolean hasAttribute(String name);

    /**
     * Returns <code>true</code> when an attribute with a given local name and
     * namespace URI is specified on this element or has a default value,
     * <code>false</code> otherwise.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value <code>null</code> as the
     * <code>namespaceURI</code> parameter for methods if they wish to have
     * no namespace.
     * @param namespaceURI The namespace URI of the attribute to look for.
     * @param localName The local name of the attribute to look for.
     * @return <code>true</code> if an attribute with the given local name
     *   and namespace URI is specified or has a default value on this
     *   element, <code>false</code> otherwise.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature <code>"XML"</code> and the language exposed
     *   through the Document does not support XML Namespaces (such as [<a href='http://www.w3.org/
         TR/1999/REC-html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public boolean hasAttributeNS(String namespaceURI,
                                  String localName)
                                  throws DOMException;

    /**
     *  The type information associated with this element.
     * @since DOM Level 3
     */
    public TypeInfo getSchemaTypeInfo();

    /**
     *  If the parameter <code>isId</code> is <code>true</code>, this method
     * declares the specified attribute to be a user-determined ID attribute
     * . This affects the value of <code>Attr.isId</code> and the behavior
     * of <code>Document.getElementById</code>, but does not change any
     * schema that may be in use, in particular this does not affect the
     * <code>Attr.schemaTypeInfo</code> of the specified <code>Attr</code>
     * node. Use the value <code>false</code> for the parameter
     * <code>isId</code> to undeclare an attribute for being a
     * user-determined ID attribute.
     * <br> To specify an attribute by local name and namespace URI, use the
     * <code>setIdAttributeNS</code> method.
     * @param name The name of the attribute.
     * @param isId Whether the attribute is a of type ID.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if the specified node is not an attribute
```

```
     *   of this element.
     * @since DOM Level 3
     */
    public void setIdAttribute(String name,
                               boolean isId)
                               throws DOMException;

    /**
     *  If the parameter <code>isId</code> is <code>true</code>, this method
     * declares the specified attribute to be a user-determined ID attribute
     * . This affects the value of <code>Attr.isId</code> and the behavior
     * of <code>Document.getElementById</code>, but does not change any
     * schema that may be in use, in particular this does not affect the
     * <code>Attr.schemaTypeInfo</code> of the specified <code>Attr</code>
     * node. Use the value <code>false</code> for the parameter
     * <code>isId</code> to undeclare an attribute for being a
     * user-determined ID attribute.
     * @param namespaceURI The namespace URI of the attribute.
     * @param localName The local name of the attribute.
     * @param isId Whether the attribute is a of type ID.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if the specified node is not an attribute
     *   of this element.
     * @since DOM Level 3
     */
    public void setIdAttributeNS(String namespaceURI,
                                 String localName,
                                 boolean isId)
                                 throws DOMException;

    /**
     *  If the parameter <code>isId</code> is <code>true</code>, this method
     * declares the specified attribute to be a user-determined ID attribute
     * . This affects the value of <code>Attr.isId</code> and the behavior
     * of <code>Document.getElementById</code>, but does not change any
     * schema that may be in use, in particular this does not affect the
     * <code>Attr.schemaTypeInfo</code> of the specified <code>Attr</code>
     * node. Use the value <code>false</code> for the parameter
     * <code>isId</code> to undeclare an attribute for being a
     * user-determined ID attribute.
     * @param idAttr The attribute node.
     * @param isId Whether the attribute is a of type ID.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if the specified node is not an attribute
     *   of this element.
     * @since DOM Level 3
     */
    public void setIdAttributeNode(Attr idAttr,
                                   boolean isId)
                                   throws DOMException;

```

```
}
```

36.18 Entity.java

Secuencia 349: org/w3c/dom/Entity.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * This interface represents a known entity, either parsed or unparsed, in an
 * XML document. Note that this models the entity itself <em>not</em> the entity declaration.
 * <p>The <code>nodeName</code> attribute that is inherited from
 * <code>Node</code> contains the name of the entity.
```

```
 * <p>An XML processor may choose to completely expand entities before the
 * structure model is passed to the DOM; in this case there will be no
 * <code>EntityReference</code> nodes in the document tree.
 * <p>XML does not mandate that a non-validating XML processor read and
 * process entity declarations made in the external subset or declared in
 * parameter entities. This means that parsed entities declared in the
 * external subset need not be expanded by some classes of applications, and
 * that the replacement text of the entity may not be available. When the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#intern-replacement'>
 * replacement text</a> is available, the corresponding <code>Entity</code> node's child list
 * represents the structure of that replacement value. Otherwise, the child
 * list is empty.
 * <p>DOM Level 3 does not support editing <code>Entity</code> nodes; if a
 * user wants to make changes to the contents of an <code>Entity</code>,
 * every related <code>EntityReference</code> node has to be replaced in the
 * structure model by a clone of the <code>Entity</code>'s contents, and
 * then the desired changes must be made to each of those clones instead.
 * <code>Entity</code> nodes and all their descendants are readonly.
 * <p>An <code>Entity</code> node does not have any parent.
 * <p ><b>Note:</b> If the entity contains an unbound namespace prefix, the
 * <code>namespaceURI</code> of the corresponding node in the
 * <code>Entity</code> node subtree is <code>null</code>. The same is true
 * for <code>EntityReference</code> nodes that refer to this entity, when
 * they are created using the <code>createEntityReference</code> method of
 * the <code>Document</code> interface.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Entity extends Node {
    /**
     * The public identifier associated with the entity if specified, and
     * <code>null</code> otherwise.
     */
    public String getPublicId();

    /**
     * The system identifier associated with the entity if specified, and
     * <code>null</code> otherwise. This may be an absolute URI or not.
     */
    public String getSystemId();

    /**
     * For unparsed entities, the name of the notation for the entity. For
     * parsed entities, this is <code>null</code>.
     */
    public String getNotationName();

    /**
     * An attribute specifying the encoding used for this entity at the time
     * of parsing, when it is an external parsed entity. This is
     * <code>null</code> if it an entity from the internal subset or if it
     * is not known.
     * @since DOM Level 3
```

```
     */
    public String getInputEncoding();

    /**
     * An attribute specifying, as part of the text declaration, the encoding
     * of this entity, when it is an external parsed entity. This is
     * <code>null</code> otherwise.
     * @since DOM Level 3
     */
    public String getXmlEncoding();

    /**
     * An attribute specifying, as part of the text declaration, the version
     * number of this entity, when it is an external parsed entity. This is
     * <code>null</code> otherwise.
     * @since DOM Level 3
     */
    public String getXmlVersion();

}
```

36.19 EntityReference.java

Secuencia 350: org/w3c/dom/EntityReference.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
```

```
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * <code>EntityReference</code> nodes may be used to represent an entity
 * reference in the tree. Note that character references and references to
 * predefined entities are considered to be expanded by the HTML or XML
 * processor so that characters are represented by their Unicode equivalent
 * rather than by an entity reference. Moreover, the XML processor may
 * completely expand references to entities while building the
 * <code>Document</code>, instead of providing <code>EntityReference</code>
 * nodes. If it does provide such nodes, then for an
 * <code>EntityReference</code> node that represents a reference to a known
 * entity an <code>Entity</code> exists, and the subtree of the
 * <code>EntityReference</code> node is a copy of the <code>Entity</code>
 * node subtree. However, the latter may not be true when an entity contains
 * an unbound namespace prefix. In such a case, because the namespace prefix
 * resolution depends on where the entity reference is, the descendants of
 * the <code>EntityReference</code> node may be bound to different namespace
 * URIs. When an <code>EntityReference</code> node represents a reference to
 * an unknown entity, the node has no children and its replacement value,
 * when used by <code>Attr.value</code> for example, is empty.
 * <p>As for <code>Entity</code> nodes, <code>EntityReference</code> nodes and
 * all their descendants are readonly.
 * <p ><b>Note:</b> <code>EntityReference</code> nodes may cause element
 * content and attribute value normalization problems when, such as in XML
 * 1.0 and XML Schema, the normalization is performed after entity reference
 * are expanded.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface EntityReference extends Node {
}
```

36.20 NameList.java

Secuencia 351: org/w3c/dom/NameList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  The <code>NameList</code> interface provides the abstraction of an ordered
 * collection of parallel pairs of name and namespace values (which could be
 * null values), without defining or constraining how this collection is
 * implemented. The items in the <code>NameList</code> are accessible via an
 * integral index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface NameList {
    /**
     *  Returns the <code>index</code>th name item in the collection.
     * @param index Index into the collection.
     * @return  The name at the <code>index</code>th position in the
```

```
     *   <code>NameList</code>, or <code>null</code> if there is no name for
     *   the specified index or if the index is out of range.
     */
    public String getName(int index);

    /**
     *  Returns the <code>index</code>th namespaceURI item in the collection.
     * @param index Index into the collection.
     * @return  The namespace URI at the <code>index</code>th position in the
     *   <code>NameList</code>, or <code>null</code> if there is no name for
     *   the specified index or if the index is out of range.
     */
    public String getNamespaceURI(int index);

    /**
     *  The number of pairs (name and namespaceURI) in the list. The range of
     * valid child node indices is 0 to <code>length-1</code> inclusive.
     */
    public int getLength();

    /**
     *  Test if a name is part of this <code>NameList</code>.
     * @param str  The name to look for.
     * @return  <code>true</code> if the name has been found,
     *   <code>false</code> otherwise.
     */
    public boolean contains(String str);

    /**
     *  Test if the pair namespaceURI/name is part of this
     * <code>NameList</code>.
     * @param namespaceURI  The namespace URI to look for.
     * @param name  The name to look for.
     * @return  <code>true</code> if the pair namespaceURI/name has been
     *   found, <code>false</code> otherwise.
     */
    public boolean containsNS(String namespaceURI,
                              String name);

}
```

36.21 NamedNodeMap.java

Secuencia 352: org/w3c/dom/NamedNodeMap.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * Objects implementing the <code>NamedNodeMap</code> interface are used to
 * represent collections of nodes that can be accessed by name. Note that
 * <code>NamedNodeMap</code> does not inherit from <code>NodeList</code>;
 * <code>NamedNodeMaps</code> are not maintained in any particular order.
 * Objects contained in an object implementing <code>NamedNodeMap</code> may
 * also be accessed by an ordinal index, but this is simply to allow
 * convenient enumeration of the contents of a <code>NamedNodeMap</code>,
 * and does not imply that the DOM specifies an order to these Nodes.
 * <p><code>NamedNodeMap</code> objects in the DOM are live.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface NamedNodeMap {
    /**
     * Retrieves a node specified by name.
     * @param name The <code>nodeName</code> of a node to retrieve.
     * @return A <code>Node</code> (of any type) with the specified
     *   <code>nodeName</code>, or <code>null</code> if it does not identify
```

```
     *   any node in this map.
     */
    public Node getNamedItem(String name);

    /**
     * Adds a node using its <code>nodeName</code> attribute. If a node with
     * that name is already present in this map, it is replaced by the new
     * one. Replacing a node by itself has no effect.
     * <br>As the <code>nodeName</code> attribute is used to derive the name
     * which the node must be stored under, multiple nodes of certain types
     * (those that have a "special" string value) cannot be stored as the
     * names would clash. This is seen as preferable to allowing nodes to be
     * aliased.
     * @param arg A node to store in this map. The node will later be
     *   accessible using the value of its <code>nodeName</code> attribute.
     * @return If the new <code>Node</code> replaces an existing node the
     *   replaced <code>Node</code> is returned, otherwise <code>null</code>
     *   is returned.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: Raised if <code>arg</code> was created from a
     *   different document than the one that created this map.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
     *   <br>INUSE_ATTRIBUTE_ERR: Raised if <code>arg</code> is an
     *   <code>Attr</code> that is already an attribute of another
     *   <code>Element</code> object. The DOM user must explicitly clone
     *   <code>Attr</code> nodes to re-use them in other elements.
     *   <br>HIERARCHY_REQUEST_ERR: Raised if an attempt is made to add a node
     *   doesn't belong in this NamedNodeMap. Examples would include trying
     *   to insert something other than an Attr node into an Element's map
     *   of attributes, or a non-Entity node into the DocumentType's map of
     *   Entities.
     */
    public Node setNamedItem(Node arg)
                            throws DOMException;

    /**
     * Removes a node specified by name. When this map contains the attributes
     * attached to an element, if the removed attribute is known to have a
     * default value, an attribute immediately appears containing the
     * default value as well as the corresponding namespace URI, local name,
     * and prefix when applicable.
     * @param name The <code>nodeName</code> of the node to remove.
     * @return The node removed from this map if a node with such a name
     *   exists.
     * @exception DOMException
     *   NOT_FOUND_ERR: Raised if there is no node named <code>name</code> in
     *   this map.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
     */
    public Node removeNamedItem(String name)
                                throws DOMException;

    /**
```

```
     * Returns the <code>index</code>th item in the map. If <code>index</code>
     * is greater than or equal to the number of nodes in this map, this
     * returns <code>null</code>.
     * @param index Index into this map.
     * @return The node at the <code>index</code>th position in the map, or
     *   <code>null</code> if that is not a valid index.
     */
    public Node item(int index);

    /**
     * The number of nodes in this map. The range of valid child node indices
     * is <code>0</code> to <code>length-1</code> inclusive.
     */
    public int getLength();

    /**
     * Retrieves a node specified by local name and namespace URI.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value null as the namespaceURI parameter
     * for methods if they wish to have no namespace.
     * @param namespaceURI The namespace URI of the node to retrieve.
     * @param localName The local name of the node to retrieve.
     * @return A <code>Node</code> (of any type) with the specified local
     *   name and namespace URI, or <code>null</code> if they do not
     *   identify any node in this map.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature "XML" and the language exposed through the
     *   Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-
         html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Node getNamedItemNS(String namespaceURI,
                               String localName)
                               throws DOMException;

    /**
     * Adds a node using its <code>namespaceURI</code> and
     * <code>localName</code>. If a node with that namespace URI and that
     * local name is already present in this map, it is replaced by the new
     * one. Replacing a node by itself has no effect.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value null as the namespaceURI parameter
     * for methods if they wish to have no namespace.
     * @param arg A node to store in this map. The node will later be
     *   accessible using the value of its <code>namespaceURI</code> and
     *   <code>localName</code> attributes.
     * @return If the new <code>Node</code> replaces an existing node the
     *   replaced <code>Node</code> is returned, otherwise <code>null</code>
     *   is returned.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: Raised if <code>arg</code> was created from a
     *   different document than the one that created this map.
```

```
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
     *   <br>INUSE_ATTRIBUTE_ERR: Raised if <code>arg</code> is an
     *   <code>Attr</code> that is already an attribute of another
     *   <code>Element</code> object. The DOM user must explicitly clone
     *   <code>Attr</code> nodes to re-use them in other elements.
     *   <br>HIERARCHY_REQUEST_ERR: Raised if an attempt is made to add a node
     *   doesn't belong in this NamedNodeMap. Examples would include trying
     *   to insert something other than an Attr node into an Element's map
     *   of attributes, or a non-Entity node into the DocumentType's map of
     *   Entities.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature "XML" and the language exposed through the
     *   Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-
         html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Node setNamedItemNS(Node arg)
                               throws DOMException;

    /**
     * Removes a node specified by local name and namespace URI. A removed
     * attribute may be known to have a default value when this map contains
     * the attributes attached to an element, as returned by the attributes
     * attribute of the <code>Node</code> interface. If so, an attribute
     * immediately appears containing the default value as well as the
     * corresponding namespace URI, local name, and prefix when applicable.
     * <br>Per [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     * , applications must use the value null as the namespaceURI parameter
     * for methods if they wish to have no namespace.
     * @param namespaceURI The namespace URI of the node to remove.
     * @param localName The local name of the node to remove.
     * @return The node removed from this map if a node with such a local
     *   name and namespace URI exists.
     * @exception DOMException
     *   NOT_FOUND_ERR: Raised if there is no node with the specified
     *   <code>namespaceURI</code> and <code>localName</code> in this map.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
     *   <br>NOT_SUPPORTED_ERR: May be raised if the implementation does not
     *   support the feature "XML" and the language exposed through the
     *   Document does not support XML Namespaces (such as [<a href='http://www.w3.org/TR/1999/REC-
         html401-19991224/'>HTML 4.01</a>]).
     * @since DOM Level 2
     */
    public Node removeNamedItemNS(String namespaceURI,
                                  String localName)
                                  throws DOMException;

}
```

36.22 Node.java

Secuencia 353: org/w3c/dom/Node.java

```
/*
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>Node</code> interface is the primary datatype for the entire
 * Document Object Model. It represents a single node in the document tree.
 * While all objects implementing the <code>Node</code> interface expose
 * methods for dealing with children, not all objects implementing the
 * <code>Node</code> interface may have children. For example,
 * <code>Text</code> nodes may not have children, and adding children to
 * such nodes results in a <code>DOMException</code> being raised.
 * <p>The attributes <code>nodeName</code>, <code>nodeValue</code> and
 * <code>attributes</code> are included as a mechanism to get at node
 * information without casting down to the specific derived interface. In
```

```
* cases where there is no obvious mapping of these attributes for a
* specific <code>nodeType</code> (e.g., <code>nodeValue</code> for an
* <code>Element</code> or <code>attributes</code> for a <code>Comment</code>
* ), this returns <code>null</code>. Note that the specialized interfaces
* may contain additional and more convenient mechanisms to get and set the
* relevant information.
* <p>The values of <code>nodeName</code>,
* <code>nodeValue</code>, and <code>attributes</code> vary according to the
* node type as follows:
* <table border='1' cellpadding='3'>
* <tr>
* <th>Interface</th>
* <th>nodeName</th>
* <th>nodeValue</th>
* <th>attributes</th>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>
* <code>Attr</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as <code>Attr.name</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as
* <code>Attr.value</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>CDATASection</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>"#cdata-section"</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as <code>CharacterData.data</code>, the
* content of the CDATA Section</td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>Comment</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>"#comment"</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as <code>CharacterData.data</code>, the
* content of the comment</td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>Document</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>"#document"</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>
* <code>DocumentFragment</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>"#document-fragment"</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>null</code></td>
```

```
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>DocumentType</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as
* <code>DocumentType.name</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'>
* <code>Element</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as <code>Element.tagName</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>NamedNodeMap</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>Entity</code></td>
* <td valign='top' rowspan='1' colspan='1'>entity name</td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>EntityReference</code></td>
* <td valign='top' rowspan='1' colspan='1'>name of entity referenced</td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>Notation</code></td>
* <td valign='top' rowspan='1' colspan='1'>notation name</td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>null</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>ProcessingInstruction</code></td>
* <td valign='top' rowspan='1' colspan='1'>same
* as <code>ProcessingInstruction.target</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as
* <code>ProcessingInstruction.data</code></td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
* </tr>
* <tr>
* <td valign='top' rowspan='1' colspan='1'><code>Text</code></td>
* <td valign='top' rowspan='1' colspan='1'>
* <code>"#text"</code></td>
* <td valign='top' rowspan='1' colspan='1'>same as <code>CharacterData.data</code>, the content
* of the text node</td>
* <td valign='top' rowspan='1' colspan='1'><code>null</code></td>
```

```
 * </tr>
 * </table>
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Node {
    // NodeType
    /**
     * The node is an <code>Element</code>.
     */
    public static final short ELEMENT_NODE              = 1;
    /**
     * The node is an <code>Attr</code>.
     */
    public static final short ATTRIBUTE_NODE            = 2;
    /**
     * The node is a <code>Text</code> node.
     */
    public static final short TEXT_NODE                 = 3;
    /**
     * The node is a <code>CDATASection</code>.
     */
    public static final short CDATA_SECTION_NODE        = 4;
    /**
     * The node is an <code>EntityReference</code>.
     */
    public static final short ENTITY_REFERENCE_NODE     = 5;
    /**
     * The node is an <code>Entity</code>.
     */
    public static final short ENTITY_NODE               = 6;
    /**
     * The node is a <code>ProcessingInstruction</code>.
     */
    public static final short PROCESSING_INSTRUCTION_NODE = 7;
    /**
     * The node is a <code>Comment</code>.
     */
    public static final short COMMENT_NODE              = 8;
    /**
     * The node is a <code>Document</code>.
     */
    public static final short DOCUMENT_NODE             = 9;
    /**
     * The node is a <code>DocumentType</code>.
     */
    public static final short DOCUMENT_TYPE_NODE        = 10;
    /**
     * The node is a <code>DocumentFragment</code>.
     */
    public static final short DOCUMENT_FRAGMENT_NODE    = 11;
    /**
     * The node is a <code>Notation</code>.
```

```
     */
    public static final short NOTATION_NODE             = 12;

    /**
     * The name of this node, depending on its type; see the table above.
     */
    public String getNodeName();

    /**
     * The value of this node, depending on its type; see the table above.
     * When it is defined to be <code>null</code>, setting it has no effect,
     * including if the node is read-only.
     * @exception DOMException
     *   DOMSTRING_SIZE_ERR: Raised when it would return more characters than
     *   fit in a <code>DOMString</code> variable on the implementation
     *   platform.
     */
    public String getNodeValue()
                            throws DOMException;
    /**
     * The value of this node, depending on its type; see the table above.
     * When it is defined to be <code>null</code>, setting it has no effect,
     * including if the node is read-only.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly and if
     *   it is not defined to be <code>null</code>.
     */
    public void setNodeValue(String nodeValue)
                            throws DOMException;

    /**
     * A code representing the type of the underlying object, as defined above.
     */
    public short getNodeType();

    /**
     * The parent of this node. All nodes, except <code>Attr</code>,
     * <code>Document</code>, <code>DocumentFragment</code>,
     * <code>Entity</code>, and <code>Notation</code> may have a parent.
     * However, if a node has just been created and not yet added to the
     * tree, or if it has been removed from the tree, this is
     * <code>null</code>.
     */
    public Node getParentNode();

    /**
     * A <code>NodeList</code> that contains all children of this node. If
     * there are no children, this is a <code>NodeList</code> containing no
     * nodes.
     */
    public NodeList getChildNodes();

    /**
```

```
     * The first child of this node. If there is no such node, this returns
     * <code>null</code>.
     */
    public Node getFirstChild();

    /**
     * The last child of this node. If there is no such node, this returns
     * <code>null</code>.
     */
    public Node getLastChild();

    /**
     * The node immediately preceding this node. If there is no such node,
     * this returns <code>null</code>.
     */
    public Node getPreviousSibling();

    /**
     * The node immediately following this node. If there is no such node,
     * this returns <code>null</code>.
     */
    public Node getNextSibling();

    /**
     * A <code>NamedNodeMap</code> containing the attributes of this node (if
     * it is an <code>Element</code>) or <code>null</code> otherwise.
     */
    public NamedNodeMap getAttributes();

    /**
     * The <code>Document</code> object associated with this node. This is
     * also the <code>Document</code> object used to create new nodes. When
     * this node is a <code>Document</code> or a <code>DocumentType</code>
     * which is not used with any <code>Document</code> yet, this is
     * <code>null</code>.
     *
     * @since DOM Level 2
     */
    public Document getOwnerDocument();

    /**
     * Inserts the node <code>newChild</code> before the existing child node
     * <code>refChild</code>. If <code>refChild</code> is <code>null</code>,
     * insert <code>newChild</code> at the end of the list of children.
     * <br>If <code>newChild</code> is a <code>DocumentFragment</code> object,
     * all of its children are inserted, in the same order, before
     * <code>refChild</code>. If the <code>newChild</code> is already in the
     * tree, it is first removed.
     * <p ><b>Note:</b>  Inserting a node before itself is implementation
     * dependent.
     * @param newChild The node to insert.
     * @param refChild The reference node, i.e., the node before which the
     *   new node must be inserted.
```

```
     * @return The node being inserted.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not
     *   allow children of the type of the <code>newChild</code> node, or if
     *   the node to insert is one of this node's ancestors or this node
     *   itself, or if this node is of type <code>Document</code> and the
     *   DOM application attempts to insert a second
     *   <code>DocumentType</code> or <code>Element</code> node.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>newChild</code> was created
     *   from a different document than the one that created this node.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly or
     *   if the parent of the node being inserted is readonly.
     *   <br>NOT_FOUND_ERR: Raised if <code>refChild</code> is not a child of
     *   this node.
     *   <br>NOT_SUPPORTED_ERR: if this node is of type <code>Document</code>,
     *   this exception might be raised if the DOM implementation doesn't
     *   support the insertion of a <code>DocumentType</code> or
     *   <code>Element</code> node.
     *
     * @since DOM Level 3
     */
    public Node insertBefore(Node newChild,
                             Node refChild)
                             throws DOMException;

    /**
     * Replaces the child node <code>oldChild</code> with <code>newChild</code>
     *  in the list of children, and returns the <code>oldChild</code> node.
     * <br>If <code>newChild</code> is a <code>DocumentFragment</code> object,
     * <code>oldChild</code> is replaced by all of the
     * <code>DocumentFragment</code> children, which are inserted in the
     * same order. If the <code>newChild</code> is already in the tree, it
     * is first removed.
     * <p ><b>Note:</b>  Replacing a node with itself is implementation
     * dependent.
     * @param newChild The new node to put in the child list.
     * @param oldChild The node being replaced in the list.
     * @return The node replaced.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not
     *   allow children of the type of the <code>newChild</code> node, or if
     *   the node to put in is one of this node's ancestors or this node
     *   itself, or if this node is of type <code>Document</code> and the
     *   result of the replacement operation would add a second
     *   <code>DocumentType</code> or <code>Element</code> on the
     *   <code>Document</code> node.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>newChild</code> was created
     *   from a different document than the one that created this node.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node or the parent of
     *   the new node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if <code>oldChild</code> is not a child of
     *   this node.
     *   <br>NOT_SUPPORTED_ERR: if this node is of type <code>Document</code>,
```

```
     *   this exception might be raised if the DOM implementation doesn't
     *   support the replacement of the <code>DocumentType</code> child or
     *   <code>Element</code> child.
     *
     * @since DOM Level 3
     */
    public Node replaceChild(Node newChild,
                             Node oldChild)
                             throws DOMException;

    /**
     * Removes the child node indicated by <code>oldChild</code> from the list
     * of children, and returns it.
     * @param oldChild The node being removed.
     * @return The node removed.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NOT_FOUND_ERR: Raised if <code>oldChild</code> is not a child of
     *   this node.
     *   <br>NOT_SUPPORTED_ERR: if this node is of type <code>Document</code>,
     *   this exception might be raised if the DOM implementation doesn't
     *   support the removal of the <code>DocumentType</code> child or the
     *   <code>Element</code> child.
     *
     * @since DOM Level 3
     */
    public Node removeChild(Node oldChild)
                            throws DOMException;

    /**
     * Adds the node <code>newChild</code> to the end of the list of children
     * of this node. If the <code>newChild</code> is already in the tree, it
     * is first removed.
     * @param newChild The node to add.If it is a
     *   <code>DocumentFragment</code> object, the entire contents of the
     *   document fragment are moved into the child list of this node
     * @return The node added.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not
     *   allow children of the type of the <code>newChild</code> node, or if
     *   the node to append is one of this node's ancestors or this node
     *   itself, or if this node is of type <code>Document</code> and the
     *   DOM application attempts to append a second
     *   <code>DocumentType</code> or <code>Element</code> node.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>newChild</code> was created
     *   from a different document than the one that created this node.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly or
     *   if the previous parent of the node being inserted is readonly.
     *   <br>NOT_SUPPORTED_ERR: if the <code>newChild</code> node is a child
     *   of the <code>Document</code> node, this exception might be raised
     *   if the DOM implementation doesn't support the removal of the
     *   <code>DocumentType</code> child or <code>Element</code> child.
     *
```

```
     * @since DOM Level 3
     */
    public Node appendChild(Node newChild)
                            throws DOMException;

    /**
     * Returns whether this node has any children.
     * @return Returns <code>true</code> if this node has any children,
     *   <code>false</code> otherwise.
     */
    public boolean hasChildNodes();

    /**
     * Returns a duplicate of this node, i.e., serves as a generic copy
     * constructor for nodes. The duplicate node has no parent (
     * <code>parentNode</code> is <code>null</code>) and no user data. User
     * data associated to the imported node is not carried over. However, if
     * any <code>UserDataHandlers</code> has been specified along with the
     * associated data these handlers will be called with the appropriate
     * parameters before this method returns.
     * <br>Cloning an <code>Element</code> copies all attributes and their
     * values, including those generated by the XML processor to represent
     * defaulted attributes, but this method does not copy any children it
     * contains unless it is a deep clone. This includes text contained in
     * an the <code>Element</code> since the text is contained in a child
     * <code>Text</code> node. Cloning an <code>Attr</code> directly, as
     * opposed to be cloned as part of an <code>Element</code> cloning
     * operation, returns a specified attribute (<code>specified</code> is
     * <code>true</code>). Cloning an <code>Attr</code> always clones its
     * children, since they represent its value, no matter whether this is a
     * deep clone or not. Cloning an <code>EntityReference</code>
     * automatically constructs its subtree if a corresponding
     * <code>Entity</code> is available, no matter whether this is a deep
     * clone or not. Cloning any other type of node simply returns a copy of
     * this node.
     * <br>Note that cloning an immutable subtree results in a mutable copy,
     * but the children of an <code>EntityReference</code> clone are readonly
     * . In addition, clones of unspecified <code>Attr</code> nodes are
     * specified. And, cloning <code>Document</code>,
     * <code>DocumentType</code>, <code>Entity</code>, and
     * <code>Notation</code> nodes is implementation dependent.
     * @param deep If <code>true</code>, recursively clone the subtree under
     *   the specified node; if <code>false</code>, clone only the node
     *   itself (and its attributes, if it is an <code>Element</code>).
     * @return The duplicate node.
     */
    public Node cloneNode(boolean deep);

    /**
     *  Puts all <code>Text</code> nodes in the full depth of the sub-tree
     * underneath this <code>Node</code>, including attribute nodes, into a
     * "normal" form where only structure (e.g., elements, comments,
     * processing instructions, CDATA sections, and entity references)
```

```
     * separates <code>Text</code> nodes, i.e., there are neither adjacent
     * <code>Text</code> nodes nor empty <code>Text</code> nodes. This can
     * be used to ensure that the DOM view of a document is the same as if
     * it were saved and re-loaded, and is useful when operations (such as
     * XPointer [<a href='http://www.w3.org/TR/2003/REC-xptr-framework-20030325/'>XPointer</a>]
     *  lookups) that depend on a particular document tree structure are to
     * be used. If the parameter "normalize-characters" of the
     * <code>DOMConfiguration</code> object attached to the
     * <code>Node.ownerDocument</code> is <code>true</code>, this method
     * will also fully normalize the characters of the <code>Text</code>
     * nodes.
     * <p ><b>Note:</b> In cases where the document contains
     * <code>CDATASections</code>, the normalize operation alone may not be
     * sufficient, since XPointers do not differentiate between
     * <code>Text</code> nodes and <code>CDATASection</code> nodes.
     *
     * @since DOM Level 3
     */
    public void normalize();

    /**
     *  Tests whether the DOM implementation implements a specific feature and
     * that feature is supported by this node, as specified in .
     * @param feature  The name of the feature to test.
     * @param version  This is the version number of the feature to test.
     * @return Returns <code>true</code> if the specified feature is
     *   supported on this node, <code>false</code> otherwise.
     *
     * @since DOM Level 2
     */
    public boolean isSupported(String feature,
                               String version);

    /**
     * The namespace URI of this node, or <code>null</code> if it is
     * unspecified (see ).
     * <br>This is not a computed value that is the result of a namespace
     * lookup based on an examination of the namespace declarations in
     * scope. It is merely the namespace URI given at creation time.
     * <br>For nodes of any type other than <code>ELEMENT_NODE</code> and
     * <code>ATTRIBUTE_NODE</code> and nodes created with a DOM Level 1
     * method, such as <code>Document.createElement()</code>, this is always
     * <code>null</code>.
     * <p ><b>Note:</b> Per the <em>Namespaces in XML</em> Specification [<a href='http://www.w3.
         org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     *  an attribute does not inherit its namespace from the element it is
     * attached to. If an attribute is not explicitly given a namespace, it
     * simply has no namespace.
     *
     * @since DOM Level 2
     */
    public String getNamespaceURI();

```

```
    /**
     * The namespace prefix of this node, or <code>null</code> if it is
     * unspecified. When it is defined to be <code>null</code>, setting it
     * has no effect, including if the node is read-only.
     * <br>Note that setting this attribute, when permitted, changes the
     * <code>nodeName</code> attribute, which holds the qualified name, as
     * well as the <code>tagName</code> and <code>name</code> attributes of
     * the <code>Element</code> and <code>Attr</code> interfaces, when
     * applicable.
     * <br>Setting the prefix to <code>null</code> makes it unspecified,
     * setting it to an empty string is implementation dependent.
     * <br>Note also that changing the prefix of an attribute that is known to
     * have a default value, does not make a new attribute with the default
     * value and the original prefix appear, since the
     * <code>namespaceURI</code> and <code>localName</code> do not change.
     * <br>For nodes of any type other than <code>ELEMENT_NODE</code> and
     * <code>ATTRIBUTE_NODE</code> and nodes created with a DOM Level 1
     * method, such as <code>createElement</code> from the
     * <code>Document</code> interface, this is always <code>null</code>.
     *
     * @since DOM Level 2
     */
    public String getPrefix();
    /**
     * The namespace prefix of this node, or <code>null</code> if it is
     * unspecified. When it is defined to be <code>null</code>, setting it
     * has no effect, including if the node is read-only.
     * <br>Note that setting this attribute, when permitted, changes the
     * <code>nodeName</code> attribute, which holds the qualified name, as
     * well as the <code>tagName</code> and <code>name</code> attributes of
     * the <code>Element</code> and <code>Attr</code> interfaces, when
     * applicable.
     * <br>Setting the prefix to <code>null</code> makes it unspecified,
     * setting it to an empty string is implementation dependent.
     * <br>Note also that changing the prefix of an attribute that is known to
     * have a default value, does not make a new attribute with the default
     * value and the original prefix appear, since the
     * <code>namespaceURI</code> and <code>localName</code> do not change.
     * <br>For nodes of any type other than <code>ELEMENT_NODE</code> and
     * <code>ATTRIBUTE_NODE</code> and nodes created with a DOM Level 1
     * method, such as <code>createElement</code> from the
     * <code>Document</code> interface, this is always <code>null</code>.
     * @exception DOMException
     *   INVALID_CHARACTER_ERR: Raised if the specified prefix contains an
     *   illegal character according to the XML version in use specified in
     *   the <code>Document.xmlVersion</code> attribute.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     *   <br>NAMESPACE_ERR: Raised if the specified <code>prefix</code> is
     *   malformed per the Namespaces in XML specification, if the
     *   <code>namespaceURI</code> of this node is <code>null</code>, if the
     *   specified prefix is "xml" and the <code>namespaceURI</code> of this
     *   node is different from "<a href='http://www.w3.org/XML/1998/namespace'>
     *   http://www.w3.org/XML/1998/namespace</a>", if this node is an attribute and the specified
```

```
        prefix is "xmlns" and
     *   the <code>namespaceURI</code> of this node is different from "<a href='http://www.w3.org
         /2000/xmlns/'>http://www.w3.org/2000/xmlns/</a>", or if this node is an attribute and the
         <code>qualifiedName</code> of
     *   this node is "xmlns" [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML
         Namespaces</a>]
     *   .
     *
     * @since DOM Level 2
     */
    public void setPrefix(String prefix)
                              throws DOMException;

    /**
     * Returns the local part of the qualified name of this node.
     * <br>For nodes of any type other than <code>ELEMENT_NODE</code> and
     * <code>ATTRIBUTE_NODE</code> and nodes created with a DOM Level 1
     * method, such as <code>Document.createElement()</code>, this is always
     * <code>null</code>.
     *
     * @since DOM Level 2
     */
    public String getLocalName();

    /**
     * Returns whether this node (if it is an element) has any attributes.
     * @return Returns <code>true</code> if this node has any attributes,
     *   <code>false</code> otherwise.
     *
     * @since DOM Level 2
     */
    public boolean hasAttributes();

    /**
     * The absolute base URI of this node or <code>null</code> if the
     * implementation wasn't able to obtain an absolute URI. This value is
     * computed as described in . However, when the <code>Document</code>
     * supports the feature "HTML" [<a href='http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML
         -20030109'>DOM Level 2 HTML</a>]
     * , the base URI is computed using first the value of the href
     * attribute of the HTML BASE element if any, and the value of the
     * <code>documentURI</code> attribute from the <code>Document</code>
     * interface otherwise.
     *
     * @since DOM Level 3
     */
    public String getBaseURI();

    // DocumentPosition
    /**
     * The two nodes are disconnected. Order between disconnected nodes is
     * always implementation-specific.
     */
```

```
    public static final short DOCUMENT_POSITION_DISCONNECTED = 0x01;
    /**
     * The second node precedes the reference node.
     */
    public static final short DOCUMENT_POSITION_PRECEDING = 0x02;
    /**
     * The node follows the reference node.
     */
    public static final short DOCUMENT_POSITION_FOLLOWING = 0x04;
    /**
     * The node contains the reference node. A node which contains is always
     * preceding, too.
     */
    public static final short DOCUMENT_POSITION_CONTAINS = 0x08;
    /**
     * The node is contained by the reference node. A node which is contained
     * is always following, too.
     */
    public static final short DOCUMENT_POSITION_CONTAINED_BY = 0x10;
    /**
     * The determination of preceding versus following is
     * implementation-specific.
     */
    public static final short DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC = 0x20;

    /**
     * Compares the reference node, i.e. the node on which this method is
     * being called, with a node, i.e. the one passed as a parameter, with
     * regard to their position in the document and according to the
     * document order.
     * @param other The node to compare against the reference node.
     * @return Returns how the node is positioned relatively to the reference
     *   node.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: when the compared nodes are from different DOM
     *   implementations that do not coordinate to return consistent
     *   implementation-specific results.
     *
     * @since DOM Level 3
     */
    public short compareDocumentPosition(Node other)
                                         throws DOMException;

    /**
     * This attribute returns the text content of this node and its
     * descendants. When it is defined to be <code>null</code>, setting it
     * has no effect. On setting, any possible children this node may have
     * are removed and, if it the new string is not empty or
     * <code>null</code>, replaced by a single <code>Text</code> node
     * containing the string this attribute is set to.
     * <br> On getting, no serialization is performed, the returned string
     * does not contain any markup. No whitespace normalization is performed
     * and the returned string does not contain the white spaces in element
```

```
     * content (see the attribute
     * <code>Text.isElementContentWhitespace</code>). Similarly, on setting,
     * no parsing is performed either, the input string is taken as pure
     * textual content.
     * <br>The string returned is made of the text content of this node
     * depending on its type, as defined below:
     * <table border='1' cellpadding='3'>
     * <tr>
     * <th>Node type</th>
     * <th>Content</th>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>
     * ELEMENT_NODE, ATTRIBUTE_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE,
     * DOCUMENT_FRAGMENT_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'>concatenation of the <code>textContent</code>
     * attribute value of every child node, excluding COMMENT_NODE and
     * PROCESSING_INSTRUCTION_NODE nodes. This is the empty string if the
     * node has no children.</td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>TEXT_NODE, CDATA_SECTION_NODE, COMMENT_NODE,
     * PROCESSING_INSTRUCTION_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'><code>nodeValue</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>DOCUMENT_NODE,
     * DOCUMENT_TYPE_NODE, NOTATION_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'><em>null</em></td>
     * </tr>
     * </table>
     * @exception DOMException
     *   DOMSTRING_SIZE_ERR: Raised when it would return more characters than
     *   fit in a <code>DOMString</code> variable on the implementation
     *   platform.
     *
     * @since DOM Level 3
     */
    public String getTextContent()
                                        throws DOMException;
    /**
     * This attribute returns the text content of this node and its
     * descendants. When it is defined to be <code>null</code>, setting it
     * has no effect. On setting, any possible children this node may have
     * are removed and, if it the new string is not empty or
     * <code>null</code>, replaced by a single <code>Text</code> node
     * containing the string this attribute is set to.
     * <br> On getting, no serialization is performed, the returned string
     * does not contain any markup. No whitespace normalization is performed
     * and the returned string does not contain the white spaces in element
     * content (see the attribute
     * <code>Text.isElementContentWhitespace</code>). Similarly, on setting,
     * no parsing is performed either, the input string is taken as pure
```

```
     * textual content.
     * <br>The string returned is made of the text content of this node
     * depending on its type, as defined below:
     * <table border='1' cellpadding='3'>
     * <tr>
     * <th>Node type</th>
     * <th>Content</th>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>
     * ELEMENT_NODE, ATTRIBUTE_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE,
     * DOCUMENT_FRAGMENT_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'>concatenation of the <code>textContent</code>
     * attribute value of every child node, excluding COMMENT_NODE and
     * PROCESSING_INSTRUCTION_NODE nodes. This is the empty string if the
     * node has no children.</td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>TEXT_NODE, CDATA_SECTION_NODE, COMMENT_NODE,
     * PROCESSING_INSTRUCTION_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'><code>nodeValue</code></td>
     * </tr>
     * <tr>
     * <td valign='top' rowspan='1' colspan='1'>DOCUMENT_NODE,
     * DOCUMENT_TYPE_NODE, NOTATION_NODE</td>
     * <td valign='top' rowspan='1' colspan='1'><em>null</em></td>
     * </tr>
     * </table>
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.
     *
     * @since DOM Level 3
     */
    public void setTextContent(String textContent)
                                        throws DOMException;

    /**
     * Returns whether this node is the same node as the given one.
     * <br>This method provides a way to determine whether two
     * <code>Node</code> references returned by the implementation reference
     * the same object. When two <code>Node</code> references are references
     * to the same object, even if through a proxy, the references may be
     * used completely interchangeably, such that all attributes have the
     * same values and calling the same DOM method on either reference
     * always has exactly the same effect.
     * @param other The node to test against.
     * @return Returns <code>true</code> if the nodes are the same,
     *   <code>false</code> otherwise.
     *
     * @since DOM Level 3
     */
    public boolean isSameNode(Node other);

```

```
    /**
     * Look up the prefix associated to the given namespace URI, starting from
     * this node. The default namespace declarations are ignored by this
     * method.
     * <br>See  for details on the algorithm used by this method.
     * @param namespaceURI The namespace URI to look for.
     * @return Returns an associated namespace prefix if found or
     *   <code>null</code> if none is found. If more than one prefix are
     *   associated to the namespace prefix, the returned namespace prefix
     *   is implementation dependent.
     *
     * @since DOM Level 3
     */
    public String lookupPrefix(String namespaceURI);

    /**
     *  This method checks if the specified <code>namespaceURI</code> is the
     * default namespace or not.
     * @param namespaceURI The namespace URI to look for.
     * @return Returns <code>true</code> if the specified
     *   <code>namespaceURI</code> is the default namespace,
     *   <code>false</code> otherwise.
     *
     * @since DOM Level 3
     */
    public boolean isDefaultNamespace(String namespaceURI);

    /**
     * Look up the namespace URI associated to the given prefix, starting from
     * this node.
     * <br>See  for details on the algorithm used by this method.
     * @param prefix The prefix to look for. If this parameter is
     *   <code>null</code>, the method will return the default namespace URI
     *   if any.
     * @return Returns the associated namespace URI or <code>null</code> if
     *   none is found.
     *
     * @since DOM Level 3
     */
    public String lookupNamespaceURI(String prefix);

    /**
     * Tests whether two nodes are equal.
     * <br>This method tests for equality of nodes, not sameness (i.e.,
     * whether the two nodes are references to the same object) which can be
     * tested with <code>Node.isSameNode()</code>. All nodes that are the
     * same will also be equal, though the reverse may not be true.
     * <br>Two nodes are equal if and only if the following conditions are
     * satisfied:
     * <ul>
     * <li>The two nodes are of the same type.
     * </li>
     * <li>The following string
```

```
     * attributes are equal: <code>nodeName</code>, <code>localName</code>,
     * <code>namespaceURI</code>, <code>prefix</code>, <code>nodeValue</code>
     * . This is: they are both <code>null</code>, or they have the same
     * length and are character for character identical.
     * </li>
     * <li>The
     * <code>attributes</code> <code>NamedNodeMaps</code> are equal. This
     * is: they are both <code>null</code>, or they have the same length and
     * for each node that exists in one map there is a node that exists in
     * the other map and is equal, although not necessarily at the same
     * index.
     * </li>
     * <li>The <code>childNodes</code> <code>NodeLists</code> are equal.
     * This is: they are both <code>null</code>, or they have the same
     * length and contain equal nodes at the same index. Note that
     * normalization can affect equality; to avoid this, nodes should be
     * normalized before being compared.
     * </li>
     * </ul>
     * <br>For two <code>DocumentType</code> nodes to be equal, the following
     * conditions must also be satisfied:
     * <ul>
     * <li>The following string attributes
     * are equal: <code>publicId</code>, <code>systemId</code>,
     * <code>internalSubset</code>.
     * </li>
     * <li>The <code>entities</code>
     * <code>NamedNodeMaps</code> are equal.
     * </li>
     * <li>The <code>notations</code>
     * <code>NamedNodeMaps</code> are equal.
     * </li>
     * </ul>
     * <br>On the other hand, the following do not affect equality: the
     * <code>ownerDocument</code>, <code>baseURI</code>, and
     * <code>parentNode</code> attributes, the <code>specified</code>
     * attribute for <code>Attr</code> nodes, the <code>schemaTypeInfo</code>
     *  attribute for <code>Attr</code> and <code>Element</code> nodes, the
     * <code>Text.isElementContentWhitespace</code> attribute for
     * <code>Text</code> nodes, as well as any user data or event listeners
     * registered on the nodes.
     * <p ><b>Note:</b>  As a general rule, anything not mentioned in the
     * description above is not significant in consideration of equality
     * checking. Note that future versions of this specification may take
     * into account more attributes and implementations conform to this
     * specification are expected to be updated accordingly.
     * @param arg The node to compare equality with.
     * @return Returns <code>true</code> if the nodes are equal,
     *   <code>false</code> otherwise.
     *
     * @since DOM Level 3
     */
    public boolean isEqualNode(Node arg);
```

```

    /**
     *  This method returns a specialized object which implements the
     * specialized APIs of the specified feature and version, as specified
     * in . The specialized object may also be obtained by using
     * binding-specific casting methods but is not necessarily expected to,
     * as discussed in . This method also allow the implementation to
     * provide specialized objects which do not support the <code>Node</code>
     *  interface.
     * @param feature  The name of the feature requested. Note that any plus
     *   sign "+" prepended to the name of the feature will be ignored since
     *   it is not significant in the context of this method.
     * @param version  This is the version number of the feature to test.
     * @return  Returns an object which implements the specialized APIs of
     *   the specified feature and version, if any, or <code>null</code> if
     *   there is no object which implements interfaces associated with that
     *   feature. If the <code>DOMObject</code> returned by this method
     *   implements the <code>Node</code> interface, it must delegate to the
     *   primary core <code>Node</code> and not return results inconsistent
     *   with the primary core <code>Node</code> such as attributes,
     *   childNodes, etc.
     *
     * @since DOM Level 3
     */
    public Object getFeature(String feature,
                             String version);

    /**
     * Associate an object to a key on this node. The object can later be
     * retrieved from this node by calling <code>getUserData</code> with the
     * same key.
     * @param key The key to associate the object to.
     * @param data The object to associate to the given key, or
     *   <code>null</code> to remove any existing association to that key.
     * @param handler The handler to associate to that key, or
     *   <code>null</code>.
     * @return Returns the <code>DOMUserData</code> previously associated to
     *   the given key on this node, or <code>null</code> if there was none.
     *
     * @since DOM Level 3
     */
    public Object setUserData(String key,
                              Object data,
                              UserDataHandler handler);

    /**
     * Retrieves the object associated to a key on a this node. The object
     * must first have been set to this node by calling
     * <code>setUserData</code> with the same key.
     * @param key The key the object is associated to.
     * @return Returns the <code>DOMUserData</code> associated to the given
     *   key on this node, or <code>null</code> if there was none.
     *
```

```
     * @since DOM Level 3
     */
    public Object getUserData(String key);

}
```

36.23 NodeList.java

Secuencia 354: org/w3c/dom/NodeList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
```

```
 * The <code>NodeList</code> interface provides the abstraction of an ordered
 * collection of nodes, without defining or constraining how this collection
 * is implemented. <code>NodeList</code> objects in the DOM are live.
 * <p>The items in the <code>NodeList</code> are accessible via an integral
 * index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface NodeList {
    /**
     * Returns the <code>index</code>th item in the collection. If
     * <code>index</code> is greater than or equal to the number of nodes in
     * the list, this returns <code>null</code>.
     * @param index Index into the collection.
     * @return The node at the <code>index</code>th position in the
     *   <code>NodeList</code>, or <code>null</code> if that is not a valid
     *   index.
     */
    public Node item(int index);

    /**
     * The number of nodes in the list. The range of valid child node indices
     * is 0 to <code>length-1</code> inclusive.
     */
    public int getLength();

}
```

36.24 Notation.java

Secuencia 355: org/w3c/dom/Notation.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * This interface represents a notation declared in the DTD. A notation either
 * declares, by name, the format of an unparsed entity (see <a href='http://www.w3.org/TR/2004/REC-xml-20040204#Notations'>section 4.7</a> of the XML 1.0 specification [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]), or is
 * used for formal declaration of processing instruction targets (see <a href='http://www.w3.org/TR/2004/REC-xml-20040204#sec-pi'>section 2.6</a> of the XML 1.0 specification [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]). The
 * <code>nodeName</code> attribute inherited from <code>Node</code> is set
 * to the declared name of the notation.
 * <p>The DOM Core does not support editing <code>Notation</code> nodes; they
 * are therefore readonly.
 * <p>A <code>Notation</code> node does not have any parent.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface Notation extends Node {
    /**
     * The public identifier of this notation. If the public identifier was
     * not specified, this is <code>null</code>.
     */
    public String getPublicId();

    /**
     * The system identifier of this notation. If the system identifier was
     * not specified, this is <code>null</code>. This may be an absolute URI
     * or not.
     */
    public String getSystemId();

}
```

36.25 ProcessingInstruction.java

Secuencia 356: org/w3c/dom/ProcessingInstruction.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>ProcessingInstruction</code> interface represents a "processing
 * instruction", used in XML as a way to keep processor-specific information
 * in the text of the document.
 * <p> No lexical check is done on the content of a processing instruction and
 * it is therefore possible to have the character sequence
 * <code>"?&gt;"</code> in the content, which is illegal a processing
```

```
 * instruction per section 2.6 of [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>]. The
 * presence of this character sequence must generate a fatal error during
 * serialization.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document Object Model (DOM) Level 3 Core Specification</a>.
 */
public interface ProcessingInstruction extends Node {
    /**
     * The target of this processing instruction. XML defines this as being
     * the first token following the markup that begins the processing
     * instruction.
     */
    public String getTarget();

    /**
     * The content of this processing instruction. This is from the first non
     * white space character after the target to the character immediately
     * preceding the <code>?&gt;</code>.
     */
    public String getData();
    /**
     * The content of this processing instruction. This is from the first non
     * white space character after the target to the character immediately
     * preceding the <code>?&gt;</code>.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.
     */
    public void setData(String data)
                                throws DOMException;

}
```

36.26 Text.java

Secuencia 357: org/w3c/dom/Text.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * The <code>Text</code> interface inherits from <code>CharacterData</code>
 * and represents the textual content (termed <a href='http://www.w3.org/TR/2004/REC-xml-20040204#
     syntax'>character data</a> in XML) of an <code>Element</code> or <code>Attr</code>. If there
     is no
 * markup inside an element's content, the text is contained in a single
 * object implementing the <code>Text</code> interface that is the only
 * child of the element. If there is markup, it is parsed into the
 * information items (elements, comments, etc.) and <code>Text</code> nodes
 * that form the list of children of the element.
 * <p>When a document is first made available via the DOM, there is only one
 * <code>Text</code> node for each block of text. Users may create adjacent
 * <code>Text</code> nodes that represent the contents of a given element
 * without any intervening markup, but should be aware that there is no way
 * to represent the separations between these nodes in XML or HTML, so they
 * will not (in general) persist between DOM editing sessions. The
 * <code>Node.normalize()</code> method merges any such adjacent
 * <code>Text</code> objects into a single node for each block of text.
 * <p> No lexical check is done on the content of a <code>Text</code> node
 * and, depending on its position in the document, some characters must be
 * escaped during serialization using character references; e.g. the
 * characters "&lt;&amp;" if the textual content is part of an element or of
 * an attribute, the character sequence "]]&gt;" when part of an element,
 * the quotation mark character " or the apostrophe character ' when part of
 * an attribute.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
```

```
 */
public interface Text extends CharacterData {
    /**
     * Breaks this node into two nodes at the specified <code>offset</code>,
     * keeping both in the tree as siblings. After being split, this node
     * will contain all the content up to the <code>offset</code> point. A
     * new node of the same type, which contains all the content at and
     * after the <code>offset</code> point, is returned. If the original
     * node had a parent node, the new node is inserted as the next sibling
     * of the original node. When the <code>offset</code> is equal to the
     * length of this node, the new node has no data.
     * @param offset The 16-bit unit offset at which to split, starting from
     *   <code>0</code>.
     * @return The new node, of the same type as this node.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified offset is negative or greater
     *   than the number of 16-bit units in <code>data</code>.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
     */
    public Text splitText(int offset)
                          throws DOMException;

    /**
     * Returns whether this text node contains <a href='http://www.w3.org/TR/2004/REC-xml-infoset
         -20040204#infoitem.character'>
     * element content whitespace</a>, often abusively called "ignorable whitespace". The text node
          is
     * determined to contain whitespace in element content during the load
     * of the document or if validation occurs while using
     * <code>Document.normalizeDocument()</code>.
     * @since DOM Level 3
     */
    public boolean isElementContentWhitespace();

    /**
     * Returns all text of <code>Text</code> nodes logically-adjacent text
     * nodes to this node, concatenated in document order.
     * <br>For instance, in the example below <code>wholeText</code> on the
     * <code>Text</code> node that contains "bar" returns "barfoo", while on
     * the <code>Text</code> node that contains "foo" it returns "barfoo".
     *
     * <pre>
     *                     +-----+
     *                     | &lt;p&gt; |
     *                     +-----+
     *                       /\
     *                      /  \
     *               /-----\    +-------+
     *               | bar |    | &amp;ent; |
     *               \-----/    +-------+
     *                               |
     *                               |
     *                          /-----\
```

```
     *                          | foo |
     *                          \-----/
     * </pre>
     * <em>Figure: barTextNode.wholeText value is "barfoo"</em>
     *
     * @since DOM Level 3
     */
    public String getWholeText();

    /**
     * Replaces the text of the current node and all logically-adjacent text
     * nodes with the specified text. All logically-adjacent text nodes are
     * removed including the current node unless it was the recipient of the
     * replacement text.
     * <p>This method returns the node which received the replacement text.
     * The returned node is:</p>
     * <ul>
     * <li><code>null</code>, when the replacement text is
     * the empty string;
     * </li>
     * <li>the current node, except when the current node is
     * read-only;
     * </li>
     * <li> a new <code>Text</code> node of the same type (
     * <code>Text</code> or <code>CDATASection</code>) as the current node
     * inserted at the location of the replacement.
     * </li>
     * </ul>
     * <p>For instance, in the above example calling
     * <code>replaceWholeText</code> on the <code>Text</code> node that
     * contains "bar" with "yo" in argument results in the following:</p>
     *
     * <pre>
     *                    +-----+
     *                    | &lt;p&gt; |
     *                    +-----+
     *                       |
     *                       |
     *                    /-----\
     *                    | yo  |
     *                    \-----/
     * </pre>
     * <em>Figure: barTextNode.replaceWholeText("yo") modifies the
     * textual content of barTextNode with "yo"</em>
     *
     * <p>Where the nodes to be removed are read-only descendants of an
     * <code>EntityReference</code>, the <code>EntityReference</code> must
     * be removed instead of the read-only nodes. If any
     * <code>EntityReference</code> to be removed has descendants that are
     * not <code>EntityReference</code>, <code>Text</code>, or
     * <code>CDATASection</code> nodes, the <code>replaceWholeText</code>
     * method must fail before performing any modification of the document,
     * raising a <code>DOMException</code> with the code
```

```
     * <code>NO_MODIFICATION_ALLOWED_ERR</code>.</p>
     * <p>For instance, in the example below calling
     * <code>replaceWholeText</code> on the <code>Text</code> node that
     * contains "bar" fails, because the <code>EntityReference</code> node
     * "ent" contains an <code>Element</code> node which cannot be removed.</p>
     * @param content The content of the replacing <code>Text</code> node.
     * @return The <code>Text</code> node created with the specified content.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if one of the <code>Text</code>
     *   nodes being replaced is readonly.
     * @since DOM Level 3
     */
    public Text replaceWholeText(String content)
                                throws DOMException;

}
```

36.27 TypeInfo.java

Secuencia 358: org/w3c/dom/TypeInfo.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
```

```
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 *  The <code>TypeInfo</code> interface represents a type referenced from
 * <code>Element</code> or <code>Attr</code> nodes, specified in the schemas
 * associated with the document. The type is a pair of a namespace URI and
 * name properties, and depends on the document's schema.
 * <p> If the document's schema is an XML DTD [<a href='http://www.w3.org/TR/2004/REC-xml
     -20040204'>XML 1.0</a>], the values
 * are computed as follows:
 * <ul>
 * <li> If this type is referenced from an
 * <code>Attr</code> node, <code>typeNamespace</code> is
 * <code>"http://www.w3.org/TR/REC-xml"</code> and <code>typeName</code>
 * represents the <b>[attribute type]</b> property in the [<a href='http://www.w3.org/TR/2004/REC-
     xml-infoset-20040204/'>XML Information Set</a>]
 * . If there is no declaration for the attribute, <code>typeNamespace</code>
 *  and <code>typeName</code> are <code>null</code>.
 * </li>
 * <li> If this type is
 * referenced from an <code>Element</code> node, <code>typeNamespace</code>
 * and <code>typeName</code> are <code>null</code>.
 * </li>
 * </ul>
 * <p> If the document's schema is an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema
     -1-20010502/'>XML Schema Part 1</a>]
 * , the values are computed as follows using the post-schema-validation
 * infoset contributions (also called PSVI contributions):
 * <ul>
 * <li> If the <b>[validity]</b> property exists AND is <em>"invalid"</em> or <em>"notKnown"</em>:
     the {target namespace} and {name} properties of the declared type if
 * available, otherwise <code>null</code>.
 * <p ><b>Note:</b>  At the time of writing, the XML Schema specification does
 * not require exposing the declared type. Thus, DOM implementations might
 * choose not to provide type information if validity is not valid.
 * </li>
 * <li> If the <b>[validity]</b> property exists and is <em>"valid"</em>:
 * <ol>
 * <li> If <b>[member type definition]</b> exists:
 * <ol>
 * <li>If {name} is not absent, then expose {name} and {target
 * namespace} properties of the <b>[member type definition]</b> property;
 * </li>
 * <li>Otherwise, expose the namespace and local name of the
 * corresponding anonymous type name.
```

```
 * </li>
 * </ol>
 * </li>
 * <li> If the <b>[type definition]</b> property exists:
 * <ol>
 * <li>If {name} is not absent, then expose {name} and {target
 * namespace} properties of the <b>[type definition]</b> property;
 * </li>
 * <li>Otherwise, expose the namespace and local name of the
 * corresponding anonymous type name.
 * </li>
 * </ol>
 * </li>
 * <li> If the <b>[member type definition anonymous]</b> exists:
 * <ol>
 * <li>If it is false, then expose <b>[member type definition name]</b> and <b>[member type
     definition namespace]</b> properties;
 * </li>
 * <li>Otherwise, expose the namespace and local name of the
 * corresponding anonymous type name.
 * </li>
 * </ol>
 * </li>
 * <li> If the <b>[type definition anonymous]</b> exists:
 * <ol>
 * <li>If it is false, then expose <b>[type definition name]</b> and <b>[type definition namespace
     ]</b> properties;
 * </li>
 * <li>Otherwise, expose the namespace and local name of the
 * corresponding anonymous type name.
 * </li>
 * </ol>
 * </li>
 * </ol>
 * </li>
 * </ul>
 * <p ><b>Note:</b>  Other schema languages are outside the scope of the W3C
 * and therefore should define how to represent their type systems using
 * <code>TypeInfo</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface TypeInfo {
    /**
     *  The name of a type declared for the associated element or attribute,
     * or <code>null</code> if unknown.
     */
    public String getTypeName();

    /**
     *  The namespace of the type declared for the associated element or
     * attribute or <code>null</code> if the element does not have
```

```
     * declaration or if no namespace information is available.
     */
    public String getTypeNamespace();

    // DerivationMethods
    /**
     *  If the document's schema is an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
     * , this constant represents the derivation by <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#key-typeRestriction'>
     * restriction</a> if complex types are involved, or a <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#element-restriction'>
     * restriction</a> if simple types are involved.
     * <br>  The reference type definition is derived by restriction from the
     * other type definition if the other type definition is the same as the
     * reference type definition, or if the other type definition can be
     * reached recursively following the {base type definition} property
     * from the reference type definition, and all the <em>derivation methods</em> involved are restriction.
     */
    public static final int DERIVATION_RESTRICTION    = 0x00000001;
    /**
     *  If the document's schema is an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
     * , this constant represents the derivation by <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#key-typeExtension'>
     * extension</a>.
     * <br>  The reference type definition is derived by extension from the
     * other type definition if the other type definition can be reached
     * recursively following the {base type definition} property from the
     * reference type definition, and at least one of the <em>derivation methods</em> involved is an extension.
     */
    public static final int DERIVATION_EXTENSION      = 0x00000002;
    /**
     *  If the document's schema is an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
     * , this constant represents the <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#element-union'>
     * union</a> if simple types are involved.
     * <br> The reference type definition is derived by union from the other
     * type definition if there exists two type definitions T1 and T2 such
     * as the reference type definition is derived from T1 by
     * <code>DERIVATION_RESTRICTION</code> or
     * <code>DERIVATION_EXTENSION</code>, T2 is derived from the other type
     * definition by <code>DERIVATION_RESTRICTION</code>, T1 has {variety} <em>union</em>, and one of the {member type definitions} is T2. Note that T1 could be
     * the same as the reference type definition, and T2 could be the same
     * as the other type definition.
     */
    public static final int DERIVATION_UNION          = 0x00000004;
    /**
     *  If the document's schema is an XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema
```

```
        -1-20010502/'>XML Schema Part 1</a>]
     * , this constant represents the <a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#
        element-list'>list</a>.
     * <br> The reference type definition is derived by list from the other
     * type definition if there exists two type definitions T1 and T2 such
     * as the reference type definition is derived from T1 by
     * <code>DERIVATION_RESTRICTION</code> or
     * <code>DERIVATION_EXTENSION</code>, T2 is derived from the other type
     * definition by <code>DERIVATION_RESTRICTION</code>, T1 has {variety} <em>list</em>, and T2 is
         the {item type definition}. Note that T1 could be the same as
     * the reference type definition, and T2 could be the same as the other
     * type definition.
     */
    public static final int DERIVATION_LIST           = 0x00000008;

    /**
     *  This method returns if there is a derivation between the reference
     * type definition, i.e. the <code>TypeInfo</code> on which the method
     * is being called, and the other type definition, i.e. the one passed
     * as parameters.
     * @param typeNamespaceArg  the namespace of the other type definition.
     * @param typeNameArg  the name of the other type definition.
     * @param derivationMethod  the type of derivation and conditions applied
     *   between two types, as described in the list of constants provided
     *   in this interface.
     * @return  If the document's schema is a DTD or no schema is associated
     *   with the document, this method will always return <code>false</code>
     *   .  If the document's schema is an XML Schema, the method will return
     *   <code>true</code> if the reference type definition is derived from
     *   the other type definition according to the derivation parameter. If
     *   the value of the parameter is <code>0</code> (no bit is set to
     *   <code>1</code> for the <code>derivationMethod</code> parameter),
     *   the method will return <code>true</code> if the other type
     *   definition can be reached by recursing any combination of {base
     *   type definition}, {item type definition}, or {member type
     *   definitions} from the reference type definition.
     */
    public boolean isDerivedFrom(String typeNamespaceArg,
                                 String typeNameArg,
                                 int derivationMethod);

}
```

36.28 UserDataHandler.java

Secuencia 359: org/w3c/dom/UserDataHandler.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom;

/**
 * When associating an object to a key on a node using
 * <code>Node.setUserData()</code> the application can provide a handler
 * that gets called when the node the object is associated to is being
 * cloned, imported, or renamed. This can be used by the application to
 * implement various behaviors regarding the data it associates to the DOM
 * nodes. This interface defines that handler.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>Document
     Object Model (DOM) Level 3 Core Specification</a>.
 * @since DOM Level 3
 */
public interface UserDataHandler {
    // OperationType
    /**
     * The node is cloned, using <code>Node.cloneNode()</code>.
     */
    public static final short NODE_CLONED               = 1;
```

```
    /**
     * The node is imported, using <code>Document.importNode()</code>.
     */
    public static final short NODE_IMPORTED            = 2;
    /**
     * The node is deleted.
     * <p ><b>Note:</b> This may not be supported or may not be reliable in
     * certain environments, such as Java, where the implementation has no
     * real control over when objects are actually deleted.
     */
    public static final short NODE_DELETED             = 3;
    /**
     * The node is renamed, using <code>Document.renameNode()</code>.
     */
    public static final short NODE_RENAMED             = 4;
    /**
     * The node is adopted, using <code>Document.adoptNode()</code>.
     */
    public static final short NODE_ADOPTED             = 5;

    /**
     * This method is called whenever the node for which this handler is
     * registered is imported or cloned.
     * <br> DOM applications must not raise exceptions in a
     * <code>UserDataHandler</code>. The effect of throwing exceptions from
     * the handler is DOM implementation dependent.
     * @param operation Specifies the type of operation that is being
     *   performed on the node.
     * @param key Specifies the key for which this handler is being called.
     * @param data Specifies the data for which this handler is being called.
     * @param src Specifies the node being cloned, adopted, imported, or
     *   renamed. This is <code>null</code> when the node is being deleted.
     * @param dst Specifies the node newly created if any, or
     *   <code>null</code>.
     */
    public void handle(short operation,
                       String key,
                       Object data,
                       Node src,
                       Node dst);

}
```

37 Dom Bootstrap (org.w3c.dom.bootstrap package)

37.1 DOMImplementationRegistry.java

Secuencia 360: org/w3c/dom/bootstrap/DOMImplementationRegistry.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */


package org.w3c.dom.bootstrap;

import java.util.StringTokenizer;
import java.util.Vector;
import org.w3c.dom.DOMImplementationSource;
import org.w3c.dom.DOMImplementationList;
import org.w3c.dom.DOMImplementation;
import java.io.InputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.security.AccessController;
import java.security.PrivilegedAction;

/**
 * A factory that enables applications to obtain instances of
 * <code>DOMImplementation</code>.
```

```
 *
 * <p>
 * Example:
 * </p>
 *
 * <pre class='example'>
 *  // get an instance of the DOMImplementation registry
 *  DOMImplementationRegistry registry =
 *       DOMImplementationRegistry.newInstance();
 *  // get a DOM implementation the Level 3 XML module
 *  DOMImplementation domImpl =
 *       registry.getDOMImplementation("XML 3.0");
 * </pre>
 *
 * <p>
 * This provides an application with an implementation-independent starting
 * point. DOM implementations may modify this class to meet new security
 * standards or to provide *additional* fallbacks for the list of
 * DOMImplementationSources.
 * </p>
 *
 * @see DOMImplementation
 * @see DOMImplementationSource
 * @since DOM Level 3
 */
public final class DOMImplementationRegistry {
    /**
     * The system property to specify the
     * DOMImplementationSource class names.
     */
    public static final String PROPERTY =
        "org.w3c.dom.DOMImplementationSourceList";

    /**
     * Default columns per line.
     */
    private static final int DEFAULT_LINE_LENGTH = 80;

    /**
     * The list of DOMImplementationSources.
     */
    private Vector sources;

    /**
     * Default class name.
     */
    private static final String FALLBACK_CLASS =
            "com.sun.org.apache.xerces.internal.dom.DOMXSImplementationSourceImpl";
    private static final String DEFAULT_PACKAGE =
            "com.sun.org.apache.xerces.internal.dom";
    /**
     * Private constructor.
     * @param srcs Vector List of DOMImplementationSources
```

```
     */
    private DOMImplementationRegistry(final Vector srcs) {
        sources = srcs;
    }

    /**
     * Obtain a new instance of a <code>DOMImplementationRegistry</code>.
     *

     * The <code>DOMImplementationRegistry</code> is initialized by the
     * application or the implementation, depending on the context, by
     * first checking the value of the Java system property
     * <code>org.w3c.dom.DOMImplementationSourceList</code> and
     * the service provider whose contents are at
     * "<code>META_INF/services/org.w3c.dom.DOMImplementationSourceList</code>".
     * The value of this property is a white-space separated list of
     * names of availables classes implementing the
     * <code>DOMImplementationSource</code> interface. Each class listed
     * in the class name list is instantiated and any exceptions
     * encountered are thrown to the application.
     *
     * @return an initialized instance of DOMImplementationRegistry
     * @throws ClassNotFoundException
     *     If any specified class can not be found
     * @throws InstantiationException
     *     If any specified class is an interface or abstract class
     * @throws IllegalAccessException
     *     If the default constructor of a specified class is not accessible
     * @throws ClassCastException
     *     If any specified class does not implement
     * <code>DOMImplementationSource</code>
     */
    public static DOMImplementationRegistry newInstance()
        throws
        ClassNotFoundException,
        InstantiationException,
        IllegalAccessException,
        ClassCastException {
        Vector sources = new Vector();

        ClassLoader classLoader = getClassLoader();
        // fetch system property:
        String p = getSystemProperty(PROPERTY);

        //
        // if property is not specified then use contents of
        // META_INF/org.w3c.dom.DOMImplementationSourceList from classpath
        if (p == null) {
            p = getServiceValue(classLoader);
        }
        if (p == null) {
            //
            // DOM Implementations can modify here to add *additional* fallback
```

```
            // mechanisms to access a list of default DOMImplementationSources.
            //fall back to JAXP implementation class com.sun.org.apache.xerces.internal.dom.
                DOMXSImplementationSourceImpl
            p = FALLBACK_CLASS;
        }
        if (p != null) {
            StringTokenizer st = new StringTokenizer(p);
            while (st.hasMoreTokens()) {
                String sourceName = st.nextToken();
                // make sure we have access to restricted packages
                boolean internal = false;
                if (System.getSecurityManager() != null) {
                    if (sourceName != null && sourceName.startsWith(DEFAULT_PACKAGE)) {
                        internal = true;
                    }
                }
                Class sourceClass = null;
                if (classLoader != null && !internal) {
                    sourceClass = classLoader.loadClass(sourceName);
                } else {
                    sourceClass = Class.forName(sourceName);
                }
                DOMImplementationSource source =
                    (DOMImplementationSource) sourceClass.newInstance();
                sources.addElement(source);
            }
        }
        return new DOMImplementationRegistry(sources);
    }

    /**
     * Return the first implementation that has the desired
     * features, or <code>null</code> if none is found.
     *
     * @param features
     *            A string that specifies which features are required. This is
     *            a space separated list in which each feature is specified by
     *            its name optionally followed by a space and a version number.
     *            This is something like: "XML 1.0 Traversal +Events 2.0"
     * @return An implementation that has the desired features,
     *         or <code>null</code> if none found.
     */
    public DOMImplementation getDOMImplementation(final String features) {
        int size = sources.size();
        String name = null;
        for (int i = 0; i < size; i++) {
            DOMImplementationSource source =
                (DOMImplementationSource) sources.elementAt(i);
            DOMImplementation impl = source.getDOMImplementation(features);
            if (impl != null) {
                return impl;
            }
        }
```

```
        return null;
    }

    /**
     * Return a list of implementations that support the
     * desired features.
     *
     * @param features
     *            A string that specifies which features are required. This is
     *            a space separated list in which each feature is specified by
     *            its name optionally followed by a space and a version number.
     *            This is something like: "XML 1.0 Traversal +Events 2.0"
     * @return A list of DOMImplementations that support the desired features.
     */
    public DOMImplementationList getDOMImplementationList(final String features) {
        final Vector implementations = new Vector();
        int size = sources.size();
        for (int i = 0; i < size; i++) {
            DOMImplementationSource source =
                (DOMImplementationSource) sources.elementAt(i);
            DOMImplementationList impls =
                source.getDOMImplementationList(features);
            for (int j = 0; j < impls.getLength(); j++) {
                DOMImplementation impl = impls.item(j);
                implementations.addElement(impl);
            }
        }
        return new DOMImplementationList() {
                public DOMImplementation item(final int index) {
                    if (index >= 0 && index < implementations.size()) {
                        try {
                            return (DOMImplementation)
                                implementations.elementAt(index);
                        } catch (ArrayIndexOutOfBoundsException e) {
                            return null;
                        }
                    }
                    return null;
                }

                public int getLength() {
                    return implementations.size();
                }
            };
    }

    /**
     * Register an implementation.
     *
     * @param s The source to be registered, may not be <code>null</code>
     */
    public void addSource(final DOMImplementationSource s) {
        if (s == null) {
```

```
            throw new NullPointerException();
        }
        if (!sources.contains(s)) {
            sources.addElement(s);
        }
    }

    /**
     *
     * Gets a class loader.
     *
     * @return A class loader, possibly <code>null</code>
     */
    private static ClassLoader getClassLoader() {
        try {
            ClassLoader contextClassLoader = getContextClassLoader();

            if (contextClassLoader != null) {
                return contextClassLoader;
            }
        } catch (Exception e) {
            // Assume that the DOM application is in a JRE 1.1, use the
            // current ClassLoader
            return DOMImplementationRegistry.class.getClassLoader();
        }
        return DOMImplementationRegistry.class.getClassLoader();
    }

    /**
     * This method attempts to return the first line of the resource
     * META_INF/services/org.w3c.dom.DOMImplementationSourceList
     * from the provided ClassLoader.
     *
     * @param classLoader classLoader, may not be <code>null</code>.
     * @return first line of resource, or <code>null</code>
     */
    private static String getServiceValue(final ClassLoader classLoader) {
        String serviceId = "META-INF/services/" + PROPERTY;
        // try to find services in CLASSPATH
        try {
            InputStream is = getResourceAsStream(classLoader, serviceId);

            if (is != null) {
                BufferedReader rd;
                try {
                    rd =
                        new BufferedReader(new InputStreamReader(is, "UTF-8"),
                                           DEFAULT_LINE_LENGTH);
                } catch (java.io.UnsupportedEncodingException e) {
                    rd =
                        new BufferedReader(new InputStreamReader(is),
                                           DEFAULT_LINE_LENGTH);
                }
```

```
                String serviceValue = rd.readLine();
                rd.close();
                if (serviceValue != null && serviceValue.length() > 0) {
                    return serviceValue;
                }
            }
        } catch (Exception ex) {
            return null;
        }
        return null;
    }

    /**
     * A simple JRE (Java Runtime Environment) 1.1 test
     *
     * @return <code>true</code> if JRE 1.1
     */
    private static boolean isJRE11() {
        try {
            Class c = Class.forName("java.security.AccessController");
            // java.security.AccessController existed since 1.2 so, if no
            // exception was thrown, the DOM application is running in a JRE
            // 1.2 or higher
            return false;
        } catch (Exception ex) {
            // ignore
        }
        return true;
    }

    /**
     * This method returns the ContextClassLoader or <code>null</code> if
     * running in a JRE 1.1
     *
     * @return The Context Classloader
     */
    private static ClassLoader getContextClassLoader() {
        return isJRE11()
            ? null
            : (ClassLoader)
              AccessController.doPrivileged(new PrivilegedAction() {
                    public Object run() {
                        ClassLoader classLoader = null;
                        try {
                            classLoader =
                                Thread.currentThread().getContextClassLoader();
                        } catch (SecurityException ex) {
                        }
                        return classLoader;
                    }
                });
    }

```

```
    /**
     * This method returns the system property indicated by the specified name
     * after checking access control privileges. For a JRE 1.1, this check is
     * not done.
     *
     * @param name the name of the system property
     * @return the system property
     */
    private static String getSystemProperty(final String name) {
        return isJRE11()
            ? (String) System.getProperty(name)
            : (String) AccessController.doPrivileged(new PrivilegedAction() {
                    public Object run() {
                        return System.getProperty(name);
                    }
                });
    }

    /**
     * This method returns an Inputstream for the reading resource
     * META_INF/services/org.w3c.dom.DOMImplementationSourceList after checking
     * access control privileges. For a JRE 1.1, this check is not done.
     *
     * @param classLoader classLoader
     * @param name the resource
     * @return an Inputstream for the resource specified
     */
    private static InputStream getResourceAsStream(final ClassLoader classLoader,
                                                   final String name) {
        if (isJRE11()) {
            InputStream ris;
            if (classLoader == null) {
                ris = ClassLoader.getSystemResourceAsStream(name);
            } else {
                ris = classLoader.getResourceAsStream(name);
            }
            return ris;
        } else {
            return (InputStream)
                AccessController.doPrivileged(new PrivilegedAction() {
                        public Object run() {
                            InputStream ris;
                            if (classLoader == null) {
                                ris =
                                    ClassLoader.getSystemResourceAsStream(name);
                            } else {
                                ris = classLoader.getResourceAsStream(name);
                            }
                            return ris;
                        }
                    });
        }
    }
```

```
}
```

38 Dom Css (org.w3c.dom.css package)

38.1 CSS2Properties.java

Secuencia 361: org/w3c/dom/css/CSS2Properties.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
```

```
 *  The <code>CSS2Properties</code> interface represents a convenience
 * mechanism for retrieving and setting properties within a
 * <code>CSSStyleDeclaration</code>. The attributes of this interface
 * correspond to all the properties specified in CSS2. Getting an attribute
 * of this interface is equivalent to calling the
 * <code>getPropertyValue</code> method of the
 * <code>CSSStyleDeclaration</code> interface. Setting an attribute of this
 * interface is equivalent to calling the <code>setProperty</code> method of
 * the <code>CSSStyleDeclaration</code> interface.
 * <p> A conformant implementation of the CSS module is not required to
 * implement the <code>CSS2Properties</code> interface. If an implementation
 * does implement this interface, the expectation is that language-specific
 * methods can be used to cast from an instance of the
 * <code>CSSStyleDeclaration</code> interface to the
 * <code>CSS2Properties</code> interface.
 * <p> If an implementation does implement this interface, it is expected to
 * understand the specific syntax of the shorthand properties, and apply
 * their semantics; when the <code>margin</code> property is set, for
 * example, the <code>marginTop</code>, <code>marginRight</code>,
 * <code>marginBottom</code> and <code>marginLeft</code> properties are
 * actually being set by the underlying implementation.
 * <p> When dealing with CSS "shorthand" properties, the shorthand properties
 * should be decomposed into their component longhand properties as
 * appropriate, and when querying for their value, the form returned should
 * be the shortest form exactly equivalent to the declarations made in the
 * ruleset. However, if there is no shorthand declaration that could be
 * added to the ruleset without changing in any way the rules already
 * declared in the ruleset (i.e., by adding longhand rules that were
 * previously not declared in the ruleset), then the empty string should be
 * returned for the shorthand property.
 * <p> For example, querying for the <code>font</code> property should not
 * return "normal normal normal 14pt/normal Arial, sans-serif", when "14pt
 * Arial, sans-serif" suffices. (The normals are initial values, and are
 * implied by use of the longhand property.)
 * <p> If the values for all the longhand properties that compose a particular
 * string are the initial values, then a string consisting of all the
 * initial values should be returned (e.g. a <code>border-width</code> value
 * of "medium" should be returned as such, not as "").
 * <p> For some shorthand properties that take missing values from other
 * sides, such as the <code>margin</code>, <code>padding</code>, and
 * <code>border-[width|style|color]</code> properties, the minimum number of
 * sides possible should be used; i.e., "0px 10px" will be returned instead
 * of "0px 10px 0px 10px".
 * <p> If the value of a shorthand property can not be decomposed into its
 * component longhand properties, as is the case for the <code>font</code>
 * property with a value of "menu", querying for the values of the component
 * longhand properties should return the empty string.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSS2Properties {
    /**
```

```
     *  See the azimuth property definition in CSS2.
     */
    public String getAzimuth();
    /**
     *  See the azimuth property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setAzimuth(String azimuth)
                                          throws DOMException;

    /**
     *  See the background property definition in CSS2.
     */
    public String getBackground();
    /**
     *  See the background property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackground(String background)
                                          throws DOMException;

    /**
     *  See the background-attachment property definition in CSS2.
     */
    public String getBackgroundAttachment();
    /**
     *  See the background-attachment property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackgroundAttachment(String backgroundAttachment)
                                          throws DOMException;

    /**
     *  See the background-color property definition in CSS2.
     */
    public String getBackgroundColor();
    /**
     *  See the background-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackgroundColor(String backgroundColor)
```

```
                                        throws DOMException;

    /**
     *  See the background-image property definition in CSS2.
     */
    public String getBackgroundImage();
    /**
     *  See the background-image property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackgroundImage(String backgroundImage)
                                        throws DOMException;

    /**
     *  See the background-position property definition in CSS2.
     */
    public String getBackgroundPosition();
    /**
     *  See the background-position property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackgroundPosition(String backgroundPosition)
                                        throws DOMException;

    /**
     *  See the background-repeat property definition in CSS2.
     */
    public String getBackgroundRepeat();
    /**
     *  See the background-repeat property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBackgroundRepeat(String backgroundRepeat)
                                        throws DOMException;

    /**
     *  See the border property definition in CSS2.
     */
    public String getBorder();
    /**
     *  See the border property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
```

```
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorder(String border)
                                          throws DOMException;

    /**
     *  See the border-collapse property definition in CSS2.
     */
    public String getBorderCollapse();
    /**
     *  See the border-collapse property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderCollapse(String borderCollapse)
                                          throws DOMException;

    /**
     *  See the border-color property definition in CSS2.
     */
    public String getBorderColor();
    /**
     *  See the border-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderColor(String borderColor)
                                          throws DOMException;

    /**
     *  See the border-spacing property definition in CSS2.
     */
    public String getBorderSpacing();
    /**
     *  See the border-spacing property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderSpacing(String borderSpacing)
                                          throws DOMException;

    /**
     *  See the border-style property definition in CSS2.
     */
    public String getBorderStyle();
    /**
     *  See the border-style property definition in CSS2.
```

```
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderStyle(String borderStyle)
                                          throws DOMException;

    /**
     *  See the border-top property definition in CSS2.
     */
    public String getBorderTop();
    /**
     *  See the border-top property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderTop(String borderTop)
                                          throws DOMException;

    /**
     *  See the border-right property definition in CSS2.
     */
    public String getBorderRight();
    /**
     *  See the border-right property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderRight(String borderRight)
                                          throws DOMException;

    /**
     *  See the border-bottom property definition in CSS2.
     */
    public String getBorderBottom();
    /**
     *  See the border-bottom property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderBottom(String borderBottom)
                                          throws DOMException;

    /**
     *  See the border-left property definition in CSS2.
     */
```

```
    public String getBorderLeft();
    /**
     *  See the border-left property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderLeft(String borderLeft)
                                          throws DOMException;

    /**
     *  See the border-top-color property definition in CSS2.
     */
    public String getBorderTopColor();
    /**
     *  See the border-top-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderTopColor(String borderTopColor)
                                          throws DOMException;

    /**
     *  See the border-right-color property definition in CSS2.
     */
    public String getBorderRightColor();
    /**
     *  See the border-right-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderRightColor(String borderRightColor)
                                          throws DOMException;

    /**
     *  See the border-bottom-color property definition in CSS2.
     */
    public String getBorderBottomColor();
    /**
     *  See the border-bottom-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderBottomColor(String borderBottomColor)
                                          throws DOMException;

```

```
    /**
     *  See the border-left-color property definition in CSS2.
     */
    public String getBorderLeftColor();
    /**
     *  See the border-left-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderLeftColor(String borderLeftColor)
                                          throws DOMException;

    /**
     *  See the border-top-style property definition in CSS2.
     */
    public String getBorderTopStyle();
    /**
     *  See the border-top-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderTopStyle(String borderTopStyle)
                                          throws DOMException;

    /**
     *  See the border-right-style property definition in CSS2.
     */
    public String getBorderRightStyle();
    /**
     *  See the border-right-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderRightStyle(String borderRightStyle)
                                          throws DOMException;

    /**
     *  See the border-bottom-style property definition in CSS2.
     */
    public String getBorderBottomStyle();
    /**
     *  See the border-bottom-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
```

```
    public void setBorderBottomStyle(String borderBottomStyle)
                                          throws DOMException;

    /**
     *  See the border-left-style property definition in CSS2.
     */
    public String getBorderLeftStyle();
    /**
     *  See the border-left-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderLeftStyle(String borderLeftStyle)
                                          throws DOMException;

    /**
     *  See the border-top-width property definition in CSS2.
     */
    public String getBorderTopWidth();
    /**
     *  See the border-top-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderTopWidth(String borderTopWidth)
                                          throws DOMException;

    /**
     *  See the border-right-width property definition in CSS2.
     */
    public String getBorderRightWidth();
    /**
     *  See the border-right-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderRightWidth(String borderRightWidth)
                                          throws DOMException;

    /**
     *  See the border-bottom-width property definition in CSS2.
     */
    public String getBorderBottomWidth();
    /**
     *  See the border-bottom-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
```

```
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderBottomWidth(String borderBottomWidth)
                                          throws DOMException;

    /**
     *  See the border-left-width property definition in CSS2.
     */
    public String getBorderLeftWidth();
    /**
     *  See the border-left-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderLeftWidth(String borderLeftWidth)
                                          throws DOMException;

    /**
     *  See the border-width property definition in CSS2.
     */
    public String getBorderWidth();
    /**
     *  See the border-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBorderWidth(String borderWidth)
                                          throws DOMException;

    /**
     *  See the bottom property definition in CSS2.
     */
    public String getBottom();
    /**
     *  See the bottom property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setBottom(String bottom)
                                          throws DOMException;

    /**
     *  See the caption-side property definition in CSS2.
     */
    public String getCaptionSide();
    /**
```

```
     *  See the caption-side property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCaptionSide(String captionSide)
                                          throws DOMException;

    /**
     *  See the clear property definition in CSS2.
     */
    public String getClear();
    /**
     *  See the clear property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setClear(String clear)
                                          throws DOMException;

    /**
     *  See the clip property definition in CSS2.
     */
    public String getClip();
    /**
     *  See the clip property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setClip(String clip)
                                          throws DOMException;

    /**
     *  See the color property definition in CSS2.
     */
    public String getColor();
    /**
     *  See the color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setColor(String color)
                                          throws DOMException;

    /**
     *  See the content property definition in CSS2.
```

```
     */
    public String getContent();
    /**
     *  See the content property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setContent(String content)
                                          throws DOMException;

    /**
     *  See the counter-increment property definition in CSS2.
     */
    public String getCounterIncrement();
    /**
     *  See the counter-increment property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCounterIncrement(String counterIncrement)
                                          throws DOMException;

    /**
     *  See the counter-reset property definition in CSS2.
     */
    public String getCounterReset();
    /**
     *  See the counter-reset property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCounterReset(String counterReset)
                                          throws DOMException;

    /**
     *  See the cue property definition in CSS2.
     */
    public String getCue();
    /**
     *  See the cue property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCue(String cue)
                                          throws DOMException;
```

```

    /**
     *  See the cue-after property definition in CSS2.
     */
    public String getCueAfter();
    /**
     *  See the cue-after property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCueAfter(String cueAfter)
                                          throws DOMException;

    /**
     *  See the cue-before property definition in CSS2.
     */
    public String getCueBefore();
    /**
     *  See the cue-before property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCueBefore(String cueBefore)
                                          throws DOMException;

    /**
     *  See the cursor property definition in CSS2.
     */
    public String getCursor();
    /**
     *  See the cursor property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCursor(String cursor)
                                          throws DOMException;

    /**
     *  See the direction property definition in CSS2.
     */
    public String getDirection();
    /**
     *  See the direction property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
```

```
     */
    public void setDirection(String direction)
                                          throws DOMException;

    /**
     *  See the display property definition in CSS2.
     */
    public String getDisplay();
    /**
     *  See the display property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setDisplay(String display)
                                          throws DOMException;

    /**
     *  See the elevation property definition in CSS2.
     */
    public String getElevation();
    /**
     *  See the elevation property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setElevation(String elevation)
                                          throws DOMException;

    /**
     *  See the empty-cells property definition in CSS2.
     */
    public String getEmptyCells();
    /**
     *  See the empty-cells property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setEmptyCells(String emptyCells)
                                          throws DOMException;

    /**
     *  See the float property definition in CSS2.
     */
    public String getCssFloat();
    /**
     *  See the float property definition in CSS2.
     * @exception DOMException
```

```
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setCssFloat(String cssFloat)
                                          throws DOMException;

    /**
     *  See the font property definition in CSS2.
     */
    public String getFont();
    /**
     *  See the font property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFont(String font)
                                          throws DOMException;

    /**
     *  See the font-family property definition in CSS2.
     */
    public String getFontFamily();
    /**
     *  See the font-family property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontFamily(String fontFamily)
                                          throws DOMException;

    /**
     *  See the font-size property definition in CSS2.
     */
    public String getFontSize();
    /**
     *  See the font-size property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontSize(String fontSize)
                                          throws DOMException;

    /**
     *  See the font-size-adjust property definition in CSS2.
     */
    public String getFontSizeAdjust();
```

```
    /**
     *  See the font-size-adjust property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontSizeAdjust(String fontSizeAdjust)
                                          throws DOMException;

    /**
     *  See the font-stretch property definition in CSS2.
     */
    public String getFontStretch();
    /**
     *  See the font-stretch property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontStretch(String fontStretch)
                                          throws DOMException;

    /**
     *  See the font-style property definition in CSS2.
     */
    public String getFontStyle();
    /**
     *  See the font-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontStyle(String fontStyle)
                                          throws DOMException;

    /**
     *  See the font-variant property definition in CSS2.
     */
    public String getFontVariant();
    /**
     *  See the font-variant property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontVariant(String fontVariant)
                                          throws DOMException;

    /**
```

```
     *  See the font-weight property definition in CSS2.
     */
    public String getFontWeight();
    /**
     *  See the font-weight property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFontWeight(String fontWeight)
                                          throws DOMException;

    /**
     *  See the height property definition in CSS2.
     */
    public String getHeight();
    /**
     *  See the height property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setHeight(String height)
                                          throws DOMException;

    /**
     *  See the left property definition in CSS2.
     */
    public String getLeft();
    /**
     *  See the left property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setLeft(String left)
                                          throws DOMException;

    /**
     *  See the letter-spacing property definition in CSS2.
     */
    public String getLetterSpacing();
    /**
     *  See the letter-spacing property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setLetterSpacing(String letterSpacing)
```

```
                                        throws DOMException;

    /**
     *  See the line-height property definition in CSS2.
     */
    public String getLineHeight();
    /**
     *  See the line-height property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setLineHeight(String lineHeight)
                                        throws DOMException;

    /**
     *  See the list-style property definition in CSS2.
     */
    public String getListStyle();
    /**
     *  See the list-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setListStyle(String listStyle)
                                        throws DOMException;

    /**
     *  See the list-style-image property definition in CSS2.
     */
    public String getListStyleImage();
    /**
     *  See the list-style-image property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setListStyleImage(String listStyleImage)
                                        throws DOMException;

    /**
     *  See the list-style-position property definition in CSS2.
     */
    public String getListStylePosition();
    /**
     *  See the list-style-position property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
```

```
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setListStylePosition(String listStylePosition)
                                            throws DOMException;

    /**
     *  See the list-style-type property definition in CSS2.
     */
    public String getListStyleType();
    /**
     *  See the list-style-type property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setListStyleType(String listStyleType)
                                            throws DOMException;

    /**
     *  See the margin property definition in CSS2.
     */
    public String getMargin();
    /**
     *  See the margin property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMargin(String margin)
                                            throws DOMException;

    /**
     *  See the margin-top property definition in CSS2.
     */
    public String getMarginTop();
    /**
     *  See the margin-top property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarginTop(String marginTop)
                                            throws DOMException;

    /**
     *  See the margin-right property definition in CSS2.
     */
    public String getMarginRight();
    /**
     *  See the margin-right property definition in CSS2.
```

```
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarginRight(String marginRight)
                                          throws DOMException;

    /**
     *  See the margin-bottom property definition in CSS2.
     */
    public String getMarginBottom();
    /**
     *  See the margin-bottom property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarginBottom(String marginBottom)
                                          throws DOMException;

    /**
     *  See the margin-left property definition in CSS2.
     */
    public String getMarginLeft();
    /**
     *  See the margin-left property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarginLeft(String marginLeft)
                                          throws DOMException;

    /**
     *  See the marker-offset property definition in CSS2.
     */
    public String getMarkerOffset();
    /**
     *  See the marker-offset property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarkerOffset(String markerOffset)
                                          throws DOMException;

    /**
     *  See the marks property definition in CSS2.
     */
```

```
    public String getMarks();
    /**
     *  See the marks property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMarks(String marks)
                                          throws DOMException;

    /**
     *  See the max-height property definition in CSS2.
     */
    public String getMaxHeight();
    /**
     *  See the max-height property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMaxHeight(String maxHeight)
                                          throws DOMException;

    /**
     *  See the max-width property definition in CSS2.
     */
    public String getMaxWidth();
    /**
     *  See the max-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMaxWidth(String maxWidth)
                                          throws DOMException;

    /**
     *  See the min-height property definition in CSS2.
     */
    public String getMinHeight();
    /**
     *  See the min-height property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMinHeight(String minHeight)
                                          throws DOMException;

```

```
    /**
     *  See the min-width property definition in CSS2.
     */
    public String getMinWidth();
    /**
     *  See the min-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setMinWidth(String minWidth)
                                          throws DOMException;

    /**
     *  See the orphans property definition in CSS2.
     */
    public String getOrphans();
    /**
     *  See the orphans property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setOrphans(String orphans)
                                          throws DOMException;

    /**
     *  See the outline property definition in CSS2.
     */
    public String getOutline();
    /**
     *  See the outline property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setOutline(String outline)
                                          throws DOMException;

    /**
     *  See the outline-color property definition in CSS2.
     */
    public String getOutlineColor();
    /**
     *  See the outline-color property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
```

```
    public void setOutlineColor(String outlineColor)
                                          throws DOMException;

    /**
     *  See the outline-style property definition in CSS2.
     */
    public String getOutlineStyle();
    /**
     *  See the outline-style property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setOutlineStyle(String outlineStyle)
                                          throws DOMException;

    /**
     *  See the outline-width property definition in CSS2.
     */
    public String getOutlineWidth();
    /**
     *  See the outline-width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setOutlineWidth(String outlineWidth)
                                          throws DOMException;

    /**
     *  See the overflow property definition in CSS2.
     */
    public String getOverflow();
    /**
     *  See the overflow property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setOverflow(String overflow)
                                          throws DOMException;

    /**
     *  See the padding property definition in CSS2.
     */
    public String getPadding();
    /**
     *  See the padding property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
```

```
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPadding(String padding)
                                          throws DOMException;

    /**
     *  See the padding-top property definition in CSS2.
     */
    public String getPaddingTop();
    /**
     *  See the padding-top property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPaddingTop(String paddingTop)
                                          throws DOMException;

    /**
     *  See the padding-right property definition in CSS2.
     */
    public String getPaddingRight();
    /**
     *  See the padding-right property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPaddingRight(String paddingRight)
                                          throws DOMException;

    /**
     *  See the padding-bottom property definition in CSS2.
     */
    public String getPaddingBottom();
    /**
     *  See the padding-bottom property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPaddingBottom(String paddingBottom)
                                          throws DOMException;

    /**
     *  See the padding-left property definition in CSS2.
     */
    public String getPaddingLeft();
    /**
```

```
     *  See the padding-left property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPaddingLeft(String paddingLeft)
                                          throws DOMException;

    /**
     *  See the page property definition in CSS2.
     */
    public String getPage();
    /**
     *  See the page property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPage(String page)
                                          throws DOMException;

    /**
     *  See the page-break-after property definition in CSS2.
     */
    public String getPageBreakAfter();
    /**
     *  See the page-break-after property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPageBreakAfter(String pageBreakAfter)
                                          throws DOMException;

    /**
     *  See the page-break-before property definition in CSS2.
     */
    public String getPageBreakBefore();
    /**
     *  See the page-break-before property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPageBreakBefore(String pageBreakBefore)
                                          throws DOMException;

    /**
     *  See the page-break-inside property definition in CSS2.
```

```
     */
    public String getPageBreakInside();
    /**
     *  See the page-break-inside property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPageBreakInside(String pageBreakInside)
                                            throws DOMException;

    /**
     *  See the pause property definition in CSS2.
     */
    public String getPause();
    /**
     *  See the pause property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPause(String pause)
                                            throws DOMException;

    /**
     *  See the pause-after property definition in CSS2.
     */
    public String getPauseAfter();
    /**
     *  See the pause-after property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPauseAfter(String pauseAfter)
                                            throws DOMException;

    /**
     *  See the pause-before property definition in CSS2.
     */
    public String getPauseBefore();
    /**
     *  See the pause-before property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPauseBefore(String pauseBefore)
                                            throws DOMException;
```

```
    /**
     *  See the pitch property definition in CSS2.
     */
    public String getPitch();
    /**
     *  See the pitch property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPitch(String pitch)
                                          throws DOMException;

    /**
     *  See the pitch-range property definition in CSS2.
     */
    public String getPitchRange();
    /**
     *  See the pitch-range property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPitchRange(String pitchRange)
                                          throws DOMException;

    /**
     *  See the play-during property definition in CSS2.
     */
    public String getPlayDuring();
    /**
     *  See the play-during property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setPlayDuring(String playDuring)
                                          throws DOMException;

    /**
     *  See the position property definition in CSS2.
     */
    public String getPosition();
    /**
     *  See the position property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
```

```
     */
    public void setPosition(String position)
                                            throws DOMException;

    /**
     *  See the quotes property definition in CSS2.
     */
    public String getQuotes();
    /**
     *  See the quotes property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setQuotes(String quotes)
                                            throws DOMException;

    /**
     *  See the richness property definition in CSS2.
     */
    public String getRichness();
    /**
     *  See the richness property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setRichness(String richness)
                                            throws DOMException;

    /**
     *  See the right property definition in CSS2.
     */
    public String getRight();
    /**
     *  See the right property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setRight(String right)
                                            throws DOMException;

    /**
     *  See the size property definition in CSS2.
     */
    public String getSize();
    /**
     *  See the size property definition in CSS2.
     * @exception DOMException
```

```
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSize(String size)
                                          throws DOMException;

    /**
     *  See the speak property definition in CSS2.
     */
    public String getSpeak();
    /**
     *  See the speak property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSpeak(String speak)
                                          throws DOMException;

    /**
     *  See the speak-header property definition in CSS2.
     */
    public String getSpeakHeader();
    /**
     *  See the speak-header property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSpeakHeader(String speakHeader)
                                          throws DOMException;

    /**
     *  See the speak-numeral property definition in CSS2.
     */
    public String getSpeakNumeral();
    /**
     *  See the speak-numeral property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSpeakNumeral(String speakNumeral)
                                          throws DOMException;

    /**
     *  See the speak-punctuation property definition in CSS2.
     */
    public String getSpeakPunctuation();
```

```
    /**
     *  See the speak-punctuation property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSpeakPunctuation(String speakPunctuation)
                                          throws DOMException;

    /**
     *  See the speech-rate property definition in CSS2.
     */
    public String getSpeechRate();
    /**
     *  See the speech-rate property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setSpeechRate(String speechRate)
                                          throws DOMException;

    /**
     *  See the stress property definition in CSS2.
     */
    public String getStress();
    /**
     *  See the stress property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setStress(String stress)
                                          throws DOMException;

    /**
     *  See the table-layout property definition in CSS2.
     */
    public String getTableLayout();
    /**
     *  See the table-layout property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTableLayout(String tableLayout)
                                          throws DOMException;

    /**
```

```
     *  See the text-align property definition in CSS2.
     */
    public String getTextAlign();
    /**
     *  See the text-align property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTextAlign(String textAlign)
                                          throws DOMException;

    /**
     *  See the text-decoration property definition in CSS2.
     */
    public String getTextDecoration();
    /**
     *  See the text-decoration property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTextDecoration(String textDecoration)
                                          throws DOMException;

    /**
     *  See the text-indent property definition in CSS2.
     */
    public String getTextIndent();
    /**
     *  See the text-indent property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTextIndent(String textIndent)
                                          throws DOMException;

    /**
     *  See the text-shadow property definition in CSS2.
     */
    public String getTextShadow();
    /**
     *  See the text-shadow property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTextShadow(String textShadow)
```

```
                                        throws DOMException;

    /**
     *  See the text-transform property definition in CSS2.
     */
    public String getTextTransform();
    /**
     *  See the text-transform property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTextTransform(String textTransform)
                                        throws DOMException;

    /**
     *  See the top property definition in CSS2.
     */
    public String getTop();
    /**
     *  See the top property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setTop(String top)
                                        throws DOMException;

    /**
     *  See the unicode-bidi property definition in CSS2.
     */
    public String getUnicodeBidi();
    /**
     *  See the unicode-bidi property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setUnicodeBidi(String unicodeBidi)
                                        throws DOMException;

    /**
     *  See the vertical-align property definition in CSS2.
     */
    public String getVerticalAlign();
    /**
     *  See the vertical-align property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
```

```
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setVerticalAlign(String verticalAlign)
                                            throws DOMException;

    /**
     *  See the visibility property definition in CSS2.
     */
    public String getVisibility();
    /**
     *  See the visibility property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setVisibility(String visibility)
                                            throws DOMException;

    /**
     *  See the voice-family property definition in CSS2.
     */
    public String getVoiceFamily();
    /**
     *  See the voice-family property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setVoiceFamily(String voiceFamily)
                                            throws DOMException;

    /**
     *  See the volume property definition in CSS2.
     */
    public String getVolume();
    /**
     *  See the volume property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setVolume(String volume)
                                            throws DOMException;

    /**
     *  See the white-space property definition in CSS2.
     */
    public String getWhiteSpace();
    /**
     *  See the white-space property definition in CSS2.
```

```
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setWhiteSpace(String whiteSpace)
                                          throws DOMException;

    /**
     *  See the widows property definition in CSS2.
     */
    public String getWidows();
    /**
     *  See the widows property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setWidows(String widows)
                                          throws DOMException;

    /**
     *  See the width property definition in CSS2.
     */
    public String getWidth();
    /**
     *  See the width property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setWidth(String width)
                                          throws DOMException;

    /**
     *  See the word-spacing property definition in CSS2.
     */
    public String getWordSpacing();
    /**
     *  See the word-spacing property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setWordSpacing(String wordSpacing)
                                          throws DOMException;

    /**
     *  See the z-index property definition in CSS2.
     */
```

```
    public String getZIndex();
    /**
     *  See the z-index property definition in CSS2.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the new value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setZIndex(String zIndex)
                                          throws DOMException;

}
```

38.2 CSSCharsetRule.java

Secuencia 362: org/w3c/dom/css/CSSCharsetRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
```

```
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSCharsetRule</code> interface represents a @charset rule in a
 * CSS style sheet. The value of the <code>encoding</code> attribute does
 * not affect the encoding of text data in the DOM objects; this encoding is
 * always UTF-16. After a stylesheet is loaded, the value of the
 * <code>encoding</code> attribute is the value found in the
 * <code>@charset</code> rule. If there was no <code>@charset</code> in the
 * original document, then no <code>CSSCharsetRule</code> is created. The
 * value of the <code>encoding</code> attribute may also be used as a hint
 * for the encoding used on serialization of the style sheet.
 * <p> The value of the @charset rule (and therefore of the
 * <code>CSSCharsetRule</code>) may not correspond to the encoding the
 * document actually came in; character encoding information e.g. in an HTTP
 * header, has priority (see CSS document representation) but this is not
 * reflected in the <code>CSSCharsetRule</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSCharsetRule extends CSSRule {
    /**
     *  The encoding information used in this <code>@charset</code> rule.
     */
    public String getEncoding();
    /**
     *  The encoding information used in this <code>@charset</code> rule.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified encoding value has a syntax error
     *   and is unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this encoding rule is
     *   readonly.
     */
    public void setEncoding(String encoding)
                            throws DOMException;

}
```

38.3 CSSFontFaceRule.java

Secuencia 363: org/w3c/dom/css/CSSFontFaceRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  The <code>CSSFontFaceRule</code> interface represents a @font-face rule in
 * a CSS style sheet. The <code>@font-face</code> rule is used to hold a set
 * of font descriptions.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSFontFaceRule extends CSSRule {
    /**
     *  The declaration-block of this rule.
     */
    public CSSStyleDeclaration getStyle();

}
```

38.4 CSSImportRule.java

Secuencia 364: org/w3c/dom/css/CSSImportRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.stylesheets.MediaList;

/**
 *  The <code>CSSImportRule</code> interface represents a @import rule within
 * a CSS style sheet. The <code>@import</code> rule is used to import style
 * rules from other style sheets.
```

```
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSImportRule extends CSSRule {
    /**
     *  The location of the style sheet to be imported. The attribute will not
     * contain the <code>"url(...)"</code> specifier around the URI.
     */
    public String getHref();

    /**
     *  A list of media types for which this style sheet may be used.
     */
    public MediaList getMedia();

    /**
     * The style sheet referred to by this rule, if it has been loaded. The
     * value of this attribute is <code>null</code> if the style sheet has
     * not yet been loaded or if it will not be loaded (e.g. if the style
     * sheet is for a media type not supported by the user agent).
     */
    public CSSStyleSheet getStyleSheet();

}
```

38.5 CSSMediaRule.java

Secuencia 365: org/w3c/dom/css/CSSMediaRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```
/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;
import org.w3c.dom.stylesheets.MediaList;

/**
 *  The <code>CSSMediaRule</code> interface represents a @media rule in a CSS
 * style sheet. A <code>@media</code> rule can be used to delimit style
 * rules for specific media types.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSMediaRule extends CSSRule {
    /**
     *  A list of media types for this rule.
     */
    public MediaList getMedia();

    /**
     *  A list of all CSS rules contained within the media block.
     */
    public CSSRuleList getCssRules();

    /**
     *  Used to insert a new rule into the media block.
     * @param rule  The parsable text representing the rule. For rule sets
     *   this contains both the selector and the style declaration. For
     *   at-rules, this specifies both the at-identifier and the rule
     *   content.
     * @param index  The index within the media block's rule collection of
     *   the rule before which to insert the specified rule. If the
     *   specified index is equal to the length of the media blocks's rule
     *   collection, the rule will be added to the end of the media block.
     * @return  The index within the media block's rule collection of the
     *   newly inserted rule.
```

```
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at the
     *   specified index, e.g., if an <code>@import</code> rule is inserted
     *   after a standard rule set or other at-rule.
     *   <br>INDEX_SIZE_ERR: Raised if the specified index is not a valid
     *   insertion point.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this media rule is
     *   readonly.
     *   <br>SYNTAX_ERR: Raised if the specified rule has a syntax error and
     *   is unparsable.
     */
    public int insertRule(String rule,
                          int index)
                          throws DOMException;

    /**
     *  Used to delete a rule from the media block.
     * @param index  The index within the media block's rule collection of
     *   the rule to remove.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified index does not correspond to
     *   a rule in the media rule list.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this media rule is
     *   readonly.
     */
    public void deleteRule(int index)
                           throws DOMException;

}
```

38.6 CSSPageRule.java

Secuencia 366: org/w3c/dom/css/CSSPageRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSPageRule</code> interface represents a @page rule within a
 * CSS style sheet. The <code>@page</code> rule is used to specify the
 * dimensions, orientation, margins, etc. of a page box for paged media.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSPageRule extends CSSRule {
    /**
     *  The parsable textual representation of the page selector for the rule.
     */
    public String getSelectorText();
    /**
     *  The parsable textual representation of the page selector for the rule.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified CSS string value has a syntax
     *   error and is unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this rule is readonly.
     */
    public void setSelectorText(String selectorText)
                            throws DOMException;

    /**
     *  The declaration-block of this rule.
     */
    public CSSStyleDeclaration getStyle();

```

```
}
```

38.7 CSSPrimitiveValue.java

Secuencia 367: org/w3c/dom/css/CSSPrimitiveValue.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSPrimitiveValue</code> interface represents a single CSS value
 * . This interface may be used to determine the value of a specific style
```

```
 * property currently set in a block or to set a specific style property
 * explicitly within the block. An instance of this interface might be
 * obtained from the <code>getPropertyCSSValue</code> method of the
 * <code>CSSStyleDeclaration</code> interface. A
 * <code>CSSPrimitiveValue</code> object only occurs in a context of a CSS
 * property.
 * <p> Conversions are allowed between absolute values (from millimeters to
 * centimeters, from degrees to radians, and so on) but not between relative
 * values. (For example, a pixel value cannot be converted to a centimeter
 * value.) Percentage values can't be converted since they are relative to
 * the parent value (or another property value). There is one exception for
 * color percentage values: since a color percentage value is relative to
 * the range 0-255, a color percentage value can be converted to a number;
 * (see also the <code>RGBColor</code> interface).
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSPrimitiveValue extends CSSValue {
    // UnitTypes
    /**
     * The value is not a recognized CSS2 value. The value can only be
     * obtained by using the <code>cssText</code> attribute.
     */
    public static final short CSS_UNKNOWN               = 0;
    /**
     * The value is a simple number. The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_NUMBER                = 1;
    /**
     * The value is a percentage. The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_PERCENTAGE            = 2;
    /**
     * The value is a length (ems). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_EMS                   = 3;
    /**
     * The value is a length (exs). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_EXS                   = 4;
    /**
     * The value is a length (px). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_PX                    = 5;
    /**
     * The value is a length (cm). The value can be obtained by using the
     * <code>getFloatValue</code> method.
```

```
     */
    public static final short CSS_CM                   = 6;
    /**
     * The value is a length (mm). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_MM                   = 7;
    /**
     * The value is a length (in). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_IN                   = 8;
    /**
     * The value is a length (pt). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_PT                   = 9;
    /**
     * The value is a length (pc). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_PC                   = 10;
    /**
     * The value is an angle (deg). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_DEG                  = 11;
    /**
     * The value is an angle (rad). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_RAD                  = 12;
    /**
     * The value is an angle (grad). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_GRAD                 = 13;
    /**
     * The value is a time (ms). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_MS                   = 14;
    /**
     * The value is a time (s). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_S                    = 15;
    /**
     * The value is a frequency (Hz). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_HZ                   = 16;
    /**
```

```
     * The value is a frequency (kHz). The value can be obtained by using the
     * <code>getFloatValue</code> method.
     */
    public static final short CSS_KHZ                   = 17;
    /**
     * The value is a number with an unknown dimension. The value can be
     * obtained by using the <code>getFloatValue</code> method.
     */
    public static final short CSS_DIMENSION             = 18;
    /**
     * The value is a STRING. The value can be obtained by using the
     * <code>getStringValue</code> method.
     */
    public static final short CSS_STRING                = 19;
    /**
     * The value is a URI. The value can be obtained by using the
     * <code>getStringValue</code> method.
     */
    public static final short CSS_URI                   = 20;
    /**
     * The value is an identifier. The value can be obtained by using the
     * <code>getStringValue</code> method.
     */
    public static final short CSS_IDENT                 = 21;
    /**
     * The value is a attribute function. The value can be obtained by using
     * the <code>getStringValue</code> method.
     */
    public static final short CSS_ATTR                  = 22;
    /**
     * The value is a counter or counters function. The value can be obtained
     * by using the <code>getCounterValue</code> method.
     */
    public static final short CSS_COUNTER               = 23;
    /**
     * The value is a rect function. The value can be obtained by using the
     * <code>getRectValue</code> method.
     */
    public static final short CSS_RECT                  = 24;
    /**
     * The value is a RGB color. The value can be obtained by using the
     * <code>getRGBColorValue</code> method.
     */
    public static final short CSS_RGBCOLOR              = 25;

    /**
     * The type of the value as defined by the constants specified above.
     */
    public short getPrimitiveType();

    /**
     *  A method to set the float value with a specified unit. If the property
     * attached with this value can not accept the specified unit or the
```

```
     * float value, the value will be unchanged and a
     * <code>DOMException</code> will be raised.
     * @param unitType  A unit code as defined above. The unit code can only
     *   be a float unit type (i.e. <code>CSS_NUMBER</code>,
     *   <code>CSS_PERCENTAGE</code>, <code>CSS_EMS</code>,
     *   <code>CSS_EXS</code>, <code>CSS_PX</code>, <code>CSS_CM</code>,
     *   <code>CSS_MM</code>, <code>CSS_IN</code>, <code>CSS_PT</code>,
     *   <code>CSS_PC</code>, <code>CSS_DEG</code>, <code>CSS_RAD</code>,
     *   <code>CSS_GRAD</code>, <code>CSS_MS</code>, <code>CSS_S</code>,
     *   <code>CSS_HZ</code>, <code>CSS_KHZ</code>,
     *   <code>CSS_DIMENSION</code>).
     * @param floatValue  The new float value.
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the attached property doesn't support
     *   the float value or the unit type.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setFloatValue(short unitType,
                              float floatValue)
                              throws DOMException;

    /**
     *  This method is used to get a float value in a specified unit. If this
     * CSS value doesn't contain a float value or can't be converted into
     * the specified unit, a <code>DOMException</code> is raised.
     * @param unitType  A unit code to get the float value. The unit code can
     *   only be a float unit type (i.e. <code>CSS_NUMBER</code>,
     *   <code>CSS_PERCENTAGE</code>, <code>CSS_EMS</code>,
     *   <code>CSS_EXS</code>, <code>CSS_PX</code>, <code>CSS_CM</code>,
     *   <code>CSS_MM</code>, <code>CSS_IN</code>, <code>CSS_PT</code>,
     *   <code>CSS_PC</code>, <code>CSS_DEG</code>, <code>CSS_RAD</code>,
     *   <code>CSS_GRAD</code>, <code>CSS_MS</code>, <code>CSS_S</code>,
     *   <code>CSS_HZ</code>, <code>CSS_KHZ</code>,
     *   <code>CSS_DIMENSION</code>).
     * @return  The float value in the specified unit.
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the CSS value doesn't contain a float
     *   value or if the float value can't be converted into the specified
     *   unit.
     */
    public float getFloatValue(short unitType)
                               throws DOMException;

    /**
     *  A method to set the string value with the specified unit. If the
     * property attached to this value can't accept the specified unit or
     * the string value, the value will be unchanged and a
     * <code>DOMException</code> will be raised.
     * @param stringType  A string code as defined above. The string code can
     *   only be a string unit type (i.e. <code>CSS_STRING</code>,
     *   <code>CSS_URI</code>, <code>CSS_IDENT</code>, and
     *   <code>CSS_ATTR</code>).
     * @param stringValue  The new string value.
```

```
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the CSS value doesn't contain a string
     *   value or if the string value can't be converted into the specified
     *   unit.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.
     */
    public void setStringValue(short stringType,
                               String stringValue)
                               throws DOMException;

    /**
     *  This method is used to get the string value. If the CSS value doesn't
     * contain a string value, a <code>DOMException</code> is raised.  Some
     * properties (like 'font-family' or 'voice-family') convert a
     * whitespace separated list of idents to a string.
     * @return  The string value in the current unit. The current
     *   <code>primitiveType</code> can only be a string unit type (i.e.
     *   <code>CSS_STRING</code>, <code>CSS_URI</code>,
     *   <code>CSS_IDENT</code> and <code>CSS_ATTR</code>).
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the CSS value doesn't contain a string
     *   value.
     */
    public String getStringValue()
                                 throws DOMException;

    /**
     *  This method is used to get the Counter value. If this CSS value
     * doesn't contain a counter value, a <code>DOMException</code> is
     * raised. Modification to the corresponding style property can be
     * achieved using the <code>Counter</code> interface.
     * @return The Counter value.
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the CSS value doesn't contain a
     *   Counter value (e.g. this is not <code>CSS_COUNTER</code>).
     */
    public Counter getCounterValue()
                                   throws DOMException;

    /**
     *  This method is used to get the Rect value. If this CSS value doesn't
     * contain a rect value, a <code>DOMException</code> is raised.
     * Modification to the corresponding style property can be achieved
     * using the <code>Rect</code> interface.
     * @return The Rect value.
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the CSS value doesn't contain a Rect
     *   value. (e.g. this is not <code>CSS_RECT</code>).
     */
    public Rect getRectValue()
                             throws DOMException;

    /**
```

```
     *  This method is used to get the RGB color. If this CSS value doesn't
     * contain a RGB color value, a <code>DOMException</code> is raised.
     * Modification to the corresponding style property can be achieved
     * using the <code>RGBColor</code> interface.
     * @return the RGB color value.
     * @exception DOMException
     *    INVALID_ACCESS_ERR: Raised if the attached property can't return a
     *   RGB color value (e.g. this is not <code>CSS_RGBCOLOR</code>).
     */
    public RGBColor getRGBColorValue()
                                    throws DOMException;

}
```

38.8 CSSRule.java

Secuencia 368: org/w3c/dom/css/CSSRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
```

```
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSRule</code> interface is the abstract base interface for any
 * type of CSS statement. This includes both rule sets and at-rules. An
 * implementation is expected to preserve all rules specified in a CSS style
 * sheet, even if the rule is not recognized by the parser. Unrecognized
 * rules are represented using the <code>CSSUnknownRule</code> interface.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSRule {
    // RuleType
    /**
     * The rule is a <code>CSSUnknownRule</code>.
     */
    public static final short UNKNOWN_RULE              = 0;
    /**
     * The rule is a <code>CSSStyleRule</code>.
     */
    public static final short STYLE_RULE                = 1;
    /**
     * The rule is a <code>CSSCharsetRule</code>.
     */
    public static final short CHARSET_RULE              = 2;
    /**
     * The rule is a <code>CSSImportRule</code>.
     */
    public static final short IMPORT_RULE               = 3;
    /**
     * The rule is a <code>CSSMediaRule</code>.
     */
    public static final short MEDIA_RULE                = 4;
    /**
     * The rule is a <code>CSSFontFaceRule</code>.
     */
    public static final short FONT_FACE_RULE            = 5;
    /**
     * The rule is a <code>CSSPageRule</code>.
     */
    public static final short PAGE_RULE                 = 6;

    /**
     *  The type of the rule, as defined above. The expectation is that
     * binding-specific casting methods can be used to cast down from an
```

```
     * instance of the <code>CSSRule</code> interface to the specific
     * derived interface implied by the <code>type</code>.
     */
    public short getType();

    /**
     *  The parsable textual representation of the rule. This reflects the
     * current state of the rule and not its initial value.
     */
    public String getCssText();
    /**
     *  The parsable textual representation of the rule. This reflects the
     * current state of the rule and not its initial value.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified CSS string value has a syntax
     *   error and is unparsable.
     *   <br>INVALID_MODIFICATION_ERR: Raised if the specified CSS string
     *   value represents a different type of rule than the current one.
     *   <br>HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at
     *   this point in the style sheet.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if the rule is readonly.
     */
    public void setCssText(String cssText)
                        throws DOMException;

    /**
     *  The style sheet that contains this rule.
     */
    public CSSStyleSheet getParentStyleSheet();

    /**
     *  If this rule is contained inside another rule (e.g. a style rule
     * inside an @media block), this is the containing rule. If this rule is
     * not nested inside any other rules, this returns <code>null</code>.
     */
    public CSSRule getParentRule();

}
```

38.9 CSSRuleList.java

Secuencia 369: org/w3c/dom/css/CSSRuleList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  The <code>CSSRuleList</code> interface provides the abstraction of an
 * ordered collection of CSS rules.
 * <p> The items in the <code>CSSRuleList</code> are accessible via an
 * integral index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSRuleList {
    /**
     *  The number of <code>CSSRules</code> in the list. The range of valid
     * child rule indices is <code>0</code> to <code>length-1</code>
     * inclusive.
     */
    public int getLength();

    /**
     *  Used to retrieve a CSS rule by ordinal index. The order in this
     * collection represents the order of the rules in the CSS style sheet.
     * If index is greater than or equal to the number of rules in the list,
```

```
     * this returns <code>null</code>.
     * @param index Index into the collection
     * @return The style rule at the <code>index</code> position in the
     *   <code>CSSRuleList</code>, or <code>null</code> if that is not a
     *   valid index.
     */
    public CSSRule item(int index);

}
```

38.10 CSSStyleDeclaration.java

Secuencia 370: org/w3c/dom/css/CSSStyleDeclaration.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */
```

```

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSStyleDeclaration</code> interface represents a single CSS
 * declaration block. This interface may be used to determine the style
 * properties currently set in a block or to set style properties explicitly
 * within the block.
 * <p> While an implementation may not recognize all CSS properties within a
 * CSS declaration block, it is expected to provide access to all specified
 * properties in the style sheet through the <code>CSSStyleDeclaration</code>
 *  interface. Furthermore, implementations that support a specific level of
 * CSS should correctly handle CSS shorthand properties for that level. For
 * a further discussion of shorthand properties, see the
 * <code>CSS2Properties</code> interface.
 * <p> This interface is also used to provide a read-only access to the
 * computed values of an element. See also the <code>ViewCSS</code>
 * interface.  The CSS Object Model doesn't provide an access to the
 * specified or actual values of the CSS cascade.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSStyleDeclaration {
    /**
     *  The parsable textual representation of the declaration block
     * (excluding the surrounding curly braces). Setting this attribute will
     * result in the parsing of the new value and resetting of all the
     * properties in the declaration block including the removal or addition
     * of properties.
     */
    public String getCssText();
    /**
     *  The parsable textual representation of the declaration block
     * (excluding the surrounding curly braces). Setting this attribute will
     * result in the parsing of the new value and resetting of all the
     * properties in the declaration block including the removal or addition
     * of properties.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified CSS string value has a syntax
     *   error and is unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is
     *   readonly or a property is readonly.
     */
    public void setCssText(String cssText)
                        throws DOMException;

    /**
     *  Used to retrieve the value of a CSS property if it has been explicitly
     * set within this declaration block.
     * @param propertyName  The name of the CSS property. See the CSS
```

```
     *   property index.
     * @return  Returns the value of the property if it has been explicitly
     *   set for this declaration block. Returns the empty string if the
     *   property has not been set.
     */
    public String getPropertyValue(String propertyName);

    /**
     *  Used to retrieve the object representation of the value of a CSS
     * property if it has been explicitly set within this declaration block.
     * This method returns <code>null</code> if the property is a shorthand
     * property. Shorthand property values can only be accessed and modified
     * as strings, using the <code>getPropertyValue</code> and
     * <code>setProperty</code> methods.
     * @param propertyName  The name of the CSS property. See the CSS
     *   property index.
     * @return  Returns the value of the property if it has been explicitly
     *   set for this declaration block. Returns <code>null</code> if the
     *   property has not been set.
     */
    public CSSValue getPropertyCSSValue(String propertyName);

    /**
     *  Used to remove a CSS property if it has been explicitly set within
     * this declaration block.
     * @param propertyName  The name of the CSS property. See the CSS
     *   property index.
     * @return  Returns the value of the property if it has been explicitly
     *   set for this declaration block. Returns the empty string if the
     *   property has not been set or the property name does not correspond
     *   to a known CSS property.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly
     *   or the property is readonly.
     */
    public String removeProperty(String propertyName)
                                 throws DOMException;

    /**
     *  Used to retrieve the priority of a CSS property (e.g. the
     * <code>"important"</code> qualifier) if the priority has been
     * explicitly set in this declaration block.
     * @param propertyName  The name of the CSS property. See the CSS
     *   property index.
     * @return  A string representing the priority (e.g.
     *   <code>"important"</code>) if the property has been explicitly set
     *   in this declaration block and has a priority specified. The empty
     *   string otherwise.
     */
    public String getPropertyPriority(String propertyName);

    /**
     *  Used to set a property value and priority within this declaration
```

```
     * block. <code>setProperty</code> permits to modify a property or add a
     * new one in the declaration block. Any call to this method may modify
     * the order of properties in the <code>item</code> method.
     * @param propertyName  The name of the CSS property. See the CSS
     *   property index.
     * @param value  The new value of the property.
     * @param priority  The new priority of the property (e.g.
     *   <code>"important"</code>) or the empty string if none.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified value has a syntax error and is
     *   unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is
     *   readonly or the property is readonly.
     */
    public void setProperty(String propertyName,
                            String value,
                            String priority)
                            throws DOMException;

    /**
     *  The number of properties that have been explicitly set in this
     * declaration block. The range of valid indices is 0 to length-1
     * inclusive.
     */
    public int getLength();

    /**
     *  Used to retrieve the properties that have been explicitly set in this
     * declaration block. The order of the properties retrieved using this
     * method does not have to be the order in which they were set. This
     * method can be used to iterate over all properties in this declaration
     * block.
     * @param index  Index of the property name to retrieve.
     * @return  The name of the property at this ordinal position. The empty
     *   string if no property exists at this position.
     */
    public String item(int index);

    /**
     *  The CSS rule that contains this declaration block or <code>null</code>
     * if this <code>CSSStyleDeclaration</code> is not attached to a
     * <code>CSSRule</code>.
     */
    public CSSRule getParentRule();

}
```

38.11 CSSStyleRule.java

Secuencia 371: org/w3c/dom/css/CSSStyleRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSStyleRule</code> interface represents a single rule set in a
 * CSS style sheet.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSStyleRule extends CSSRule {
    /**
     *  The textual representation of the selector for the rule set. The
     * implementation may have stripped out insignificant whitespace while
```

```
     * parsing the selector.
     */
    public String getSelectorText();
    /**
     *  The textual representation of the selector for the rule set. The
     * implementation may have stripped out insignificant whitespace while
     * parsing the selector.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified CSS string value has a syntax
     *   error and is unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this rule is readonly.
     */
    public void setSelectorText(String selectorText)
                        throws DOMException;

    /**
     *  The declaration-block of this rule set.
     */
    public CSSStyleDeclaration getStyle();

}
```

38.12 CSSStyleSheet.java

Secuencia 372: org/w3c/dom/css/CSSStyleSheet.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;
import org.w3c.dom.stylesheets.StyleSheet;

/**
 *  The <code>CSSStyleSheet</code> interface is a concrete interface used to
 * represent a CSS style sheet i.e., a style sheet whose content type is
 * "text/css".
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSStyleSheet extends StyleSheet {
    /**
     *  If this style sheet comes from an <code>@import</code> rule, the
     * <code>ownerRule</code> attribute will contain the
     * <code>CSSImportRule</code>. In that case, the <code>ownerNode</code>
     * attribute in the <code>StyleSheet</code> interface will be
     * <code>null</code>. If the style sheet comes from an element or a
     * processing instruction, the <code>ownerRule</code> attribute will be
     * <code>null</code> and the <code>ownerNode</code> attribute will
     * contain the <code>Node</code>.
     */
    public CSSRule getOwnerRule();

    /**
     *  The list of all CSS rules contained within the style sheet. This
     * includes both rule sets and at-rules.
     */
    public CSSRuleList getCssRules();

    /**
     *  Used to insert a new rule into the style sheet. The new rule now
     * becomes part of the cascade.
     * @param rule  The parsable text representing the rule. For rule sets
     *   this contains both the selector and the style declaration. For
     *   at-rules, this specifies both the at-identifier and the rule
     *   content.
     * @param index  The index within the style sheet's rule list of the rule
```

```
     *   before which to insert the specified rule. If the specified index
     *   is equal to the length of the style sheet's rule collection, the
     *   rule will be added to the end of the style sheet.
     * @return  The index within the style sheet's rule collection of the
     *   newly inserted rule.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at the
     *   specified index e.g. if an <code>@import</code> rule is inserted
     *   after a standard rule set or other at-rule.
     *   <br>INDEX_SIZE_ERR: Raised if the specified index is not a valid
     *   insertion point.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is
     *   readonly.
     *   <br>SYNTAX_ERR: Raised if the specified rule has a syntax error and
     *   is unparsable.
     */
    public int insertRule(String rule,
                          int index)
                          throws DOMException;

    /**
     *  Used to delete a rule from the style sheet.
     * @param index  The index within the style sheet's rule list of the rule
     *   to remove.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if the specified index does not correspond to
     *   a rule in the style sheet's rule list.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is
     *   readonly.
     */
    public void deleteRule(int index)
                           throws DOMException;

}
```

38.13 CSSUnknownRule.java

Secuencia 373: org/w3c/dom/css/CSSUnknownRule.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  The <code>CSSUnknownRule</code> interface represents an at-rule not
 * supported by this user agent.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSUnknownRule extends CSSRule {
}
```

38.14 CSSValue.java

Secuencia 374: org/w3c/dom/css/CSSValue.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

/**
 *  The <code>CSSValue</code> interface represents a simple or a complex
 * value. A <code>CSSValue</code> object only occurs in a context of a CSS
 * property.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSValue {
    // UnitTypes
    /**
     * The value is inherited and the <code>cssText</code> contains "inherit".
     */
    public static final short CSS_INHERIT               = 0;
    /**
     * The value is a primitive value and an instance of the
     * <code>CSSPrimitiveValue</code> interface can be obtained by using
     * binding-specific casting methods on this instance of the
     * <code>CSSValue</code> interface.
     */
```

```
    public static final short CSS_PRIMITIVE_VALUE       = 1;
    /**
     * The value is a <code>CSSValue</code> list and an instance of the
     * <code>CSSValueList</code> interface can be obtained by using
     * binding-specific casting methods on this instance of the
     * <code>CSSValue</code> interface.
     */
    public static final short CSS_VALUE_LIST            = 2;
    /**
     * The value is a custom value.
     */
    public static final short CSS_CUSTOM                = 3;

    /**
     *  A string representation of the current value.
     */
    public String getCssText();
    /**
     *  A string representation of the current value.
     * @exception DOMException
     *    SYNTAX_ERR: Raised if the specified CSS string value has a syntax
     *   error (according to the attached property) or is unparsable.
     *   <br>INVALID_MODIFICATION_ERR: Raised if the specified CSS string
     *   value represents a different type of values than the values allowed
     *   by the CSS property.
     *   <br> NO_MODIFICATION_ALLOWED_ERR: Raised if this value is readonly.
     */
    public void setCssText(String cssText)
                        throws DOMException;

    /**
     *  A code defining the type of the value as defined above.
     */
    public short getCssValueType();

}
```

38.15 CSSValueList.java

Secuencia 375: org/w3c/dom/css/CSSValueList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 * The <code>CSSValueList</code> interface provides the abstraction of an
 * ordered collection of CSS values.
 * <p> Some properties allow an empty list into their syntax. In that case,
 * these properties take the <code>none</code> identifier. So, an empty list
 * means that the property has the value <code>none</code>.
 * <p> The items in the <code>CSSValueList</code> are accessible via an
 * integral index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface CSSValueList extends CSSValue {
    /**
     * The number of <code>CSSValues</code> in the list. The range of valid
     * values of the indices is <code>0</code> to <code>length-1</code>
     * inclusive.
     */
    public int getLength();

    /**
     * Used to retrieve a <code>CSSValue</code> by ordinal index. The order in
     * this collection represents the order of the values in the CSS style
```

```
     * property. If index is greater than or equal to the number of values
     * in the list, this returns <code>null</code>.
     * @param index Index into the collection.
     * @return The <code>CSSValue</code> at the <code>index</code> position
     *   in the <code>CSSValueList</code>, or <code>null</code> if that is
     *   not a valid index.
     */
    public CSSValue item(int index);

}
```

38.16 Counter.java

Secuencia 376: org/w3c/dom/css/Counter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
```

```
 */

package org.w3c.dom.css;

/**
 *  The <code>Counter</code> interface is used to represent any counter or
 * counters function value. This interface reflects the values in the
 * underlying style property.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface Counter {
    /**
     *  This attribute is used for the identifier of the counter.
     */
    public String getIdentifier();

    /**
     *  This attribute is used for the style of the list.
     */
    public String getListStyle();

    /**
     *  This attribute is used for the separator of the nested counters.
     */
    public String getSeparator();

}
```

38.17 DOMImplementationCSS.java

Secuencia 377: org/w3c/dom/css/DOMImplementationCSS.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.DOMImplementation;
import org.w3c.dom.DOMException;

/**
 *  This interface allows the DOM user to create a <code>CSSStyleSheet</code>
 * outside the context of a document. There is no way to associate the new
 * <code>CSSStyleSheet</code> with a document in DOM Level 2.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM   Level 2
 */
public interface DOMImplementationCSS extends DOMImplementation {
    /**
     * Creates a new <code>CSSStyleSheet</code>.
     * @param title  The advisory title. See also the  section.
     * @param media  The comma-separated list of media associated with the
     *   new style sheet. See also the  section.
     * @return A new CSS style sheet.
     * @exception DOMException
     *    SYNTAX_ERR: Raised if the specified media string value has a syntax
     *   error and is unparsable.
     */
    public CSSStyleSheet createCSSStyleSheet(String title,
                                             String media)
                                             throws DOMException;

}
```

38.18 DocumentCSS.java

Secuencia 378: org/w3c/dom/css/DocumentCSS.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.Element;
import org.w3c.dom.stylesheets.DocumentStyle;

/**
 * This interface represents a document with a CSS view.
 * <p> The <code>getOverrideStyle</code> method provides a mechanism through
 * which a DOM author could effect immediate change to the style of an
 * element without modifying the explicitly linked style sheets of a
 * document or the inline style of elements in the style sheets. This style
```

```
 * sheet comes after the author style sheet in the cascade algorithm and is
 * called override style sheet. The override style sheet takes precedence
 * over author style sheets. An "!important" declaration still takes
 * precedence over a normal declaration. Override, author, and user style
 * sheets all may contain "!important" declarations. User "!important" rules
 * take precedence over both override and author "!important" rules, and
 * override "!important" rules take precedence over author "!important"
 * rules.
 * <p> The expectation is that an instance of the <code>DocumentCSS</code>
 * interface can be obtained by using binding-specific casting methods on an
 * instance of the <code>Document</code> interface.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface DocumentCSS extends DocumentStyle {
    /**
     *  This method is used to retrieve the override style declaration for a
     * specified element and a specified pseudo-element.
     * @param elt  The element whose style is to be modified. This parameter
     *   cannot be null.
     * @param pseudoElt  The pseudo-element or <code>null</code> if none.
     * @return  The override style declaration.
     */
    public CSSStyleDeclaration getOverrideStyle(Element elt,
                                                String pseudoElt);

}
```

38.19 ElementCSSInlineStyle.java

Secuencia 379: org/w3c/dom/css/ElementCSSInlineStyle.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  Inline style information attached to elements is exposed through the
 * <code>style</code> attribute. This represents the contents of the STYLE
 * attribute for HTML elements (or elements in other schemas or DTDs which
 * use the STYLE attribute in the same way). The expectation is that an
 * instance of the ElementCSSInlineStyle interface can be obtained by using
 * binding-specific casting methods on an instance of the Element interface
 * when the element supports inline CSS style informations.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface ElementCSSInlineStyle {
    /**
     *  The style attribute.
     */
    public CSSStyleDeclaration getStyle();

}
```

38.20 RGBColor.java

Secuencia 380: org/w3c/dom/css/RGBColor.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  The <code>RGBColor</code> interface is used to represent any RGB color
 * value. This interface reflects the values in the underlying style
 * property. Hence, modifications made to the <code>CSSPrimitiveValue</code>
 * objects modify the style property.
 * <p> A specified RGB color is not clipped (even if the number is outside the
 * range 0-255 or 0%-100%). A computed RGB color is clipped depending on the
 * device.
 * <p> Even if a style sheet can only contain an integer for a color value,
 * the internal storage of this integer is a float, and this can be used as
 * a float in the specified or the computed style.
 * <p> A color percentage value can always be converted to a number and vice
 * versa.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface RGBColor {
```

```
    /**
     *  This attribute is used for the red value of the RGB color.
     */
    public CSSPrimitiveValue getRed();

    /**
     *  This attribute is used for the green value of the RGB color.
     */
    public CSSPrimitiveValue getGreen();

    /**
     *  This attribute is used for the blue value of the RGB color.
     */
    public CSSPrimitiveValue getBlue();

}
```

38.21 Rect.java

Secuencia 381: org/w3c/dom/css/Rect.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
```

```
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

/**
 *  The <code>Rect</code> interface is used to represent any rect value. This
 * interface reflects the values in the underlying style property. Hence,
 * modifications made to the <code>CSSPrimitiveValue</code> objects modify
 * the style property.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface Rect {
    /**
     *  This attribute is used for the top of the rect.
     */
    public CSSPrimitiveValue getTop();

    /**
     *  This attribute is used for the right of the rect.
     */
    public CSSPrimitiveValue getRight();

    /**
     *  This attribute is used for the bottom of the rect.
     */
    public CSSPrimitiveValue getBottom();

    /**
     *  This attribute is used for the left of the rect.
     */
    public CSSPrimitiveValue getLeft();

}
```

38.22 ViewCSS.java

Secuencia 382: org/w3c/dom/css/ViewCSS.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.css;

import org.w3c.dom.Element;
import org.w3c.dom.views.AbstractView;

/**
 *  This interface represents a CSS view. The <code>getComputedStyle</code>
 * method provides a read only access to the computed values of an element.
 * <p> The expectation is that an instance of the <code>ViewCSS</code>
 * interface can be obtained by using binding-specific casting methods on an
 * instance of the <code>AbstractView</code> interface.
 * <p> Since a computed style is related to an <code>Element</code> node, if
 * this element is removed from the document, the associated
 * <code>CSSStyleDeclaration</code> and <code>CSSValue</code> related to
 * this declaration are no longer valid.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface ViewCSS extends AbstractView {
```

```
    /**
     *  This method is used to get the computed style as it is defined in [<a href='http://www.w3.
         org/TR/1998/REC-CSS2-19980512'>CSS2</a>].
     * @param elt  The element whose style is to be computed. This parameter
     *   cannot be null.
     * @param pseudoElt  The pseudo-element or <code>null</code> if none.
     * @return  The computed style. The <code>CSSStyleDeclaration</code> is
     *   read-only and contains only absolute values.
     */
    public CSSStyleDeclaration getComputedStyle(Element elt,
                                                String pseudoElt);

}
```

39 Dom Events (org.w3c.dom.events package)

39.1 DocumentEvent.java

Secuencia 383: org/w3c/dom/events/DocumentEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
```

```
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

import org.w3c.dom.DOMException;

/**
 *  The <code>DocumentEvent</code> interface provides a mechanism by which the
 * user can create an Event of a type supported by the implementation. It is
 * expected that the <code>DocumentEvent</code> interface will be
 * implemented on the same object which implements the <code>Document</code>
 * interface in an implementation which supports the Event model.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface DocumentEvent {
    /**
     *
     * @param eventType The <code>eventType</code> parameter specifies the
     *   type of <code>Event</code> interface to be created. If the
     *   <code>Event</code> interface specified is supported by the
     *   implementation this method will return a new <code>Event</code> of
     *   the interface type requested. If the <code>Event</code> is to be
     *   dispatched via the <code>dispatchEvent</code> method the
     *   appropriate event init method must be called after creation in
     *   order to initialize the <code>Event</code>'s values. As an example,
     *   a user wishing to synthesize some kind of <code>UIEvent</code>
     *   would call <code>createEvent</code> with the parameter "UIEvents".
     *   The <code>initUIEvent</code> method could then be called on the
     *   newly created <code>UIEvent</code> to set the specific type of
     *   UIEvent to be dispatched and set its context information.The
     *   <code>createEvent</code> method is used in creating
     *   <code>Event</code>s when it is either inconvenient or unnecessary
     *   for the user to create an <code>Event</code> themselves. In cases
     *   where the implementation provided <code>Event</code> is
     *   insufficient, users may supply their own <code>Event</code>
     *   implementations for use with the <code>dispatchEvent</code> method.
     * @return The newly created <code>Event</code>
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if the implementation does not support the
     *   type of <code>Event</code> interface requested
     */
    public Event createEvent(String eventType)
                             throws DOMException;

}
```

39.2 Event.java

Secuencia 384: org/w3c/dom/events/Event.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

/**
 * The <code>Event</code> interface is used to provide contextual information
 * about an event to the handler processing the event. An object which
 * implements the <code>Event</code> interface is generally passed as the
 * first parameter to an event handler. More specific context information is
 * passed to event handlers by deriving additional interfaces from
 * <code>Event</code> which contain information directly relating to the
```

```
 * type of event they accompany. These derived interfaces are also
 * implemented by the object passed to the event listener.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface Event {
    // PhaseType
    /**
     * The current event phase is the capturing phase.
     */
    public static final short CAPTURING_PHASE          = 1;
    /**
     * The event is currently being evaluated at the target
     * <code>EventTarget</code>.
     */
    public static final short AT_TARGET                = 2;
    /**
     * The current event phase is the bubbling phase.
     */
    public static final short BUBBLING_PHASE           = 3;

    /**
     * The name of the event (case-insensitive). The name must be an XML name.
     */
    public String getType();

    /**
     * Used to indicate the <code>EventTarget</code> to which the event was
     * originally dispatched.
     */
    public EventTarget getTarget();

    /**
     * Used to indicate the <code>EventTarget</code> whose
     * <code>EventListeners</code> are currently being processed. This is
     * particularly useful during capturing and bubbling.
     */
    public EventTarget getCurrentTarget();

    /**
     * Used to indicate which phase of event flow is currently being
     * evaluated.
     */
    public short getEventPhase();

    /**
     * Used to indicate whether or not an event is a bubbling event. If the
     * event can bubble the value is true, else the value is false.
     */
    public boolean getBubbles();

    /**
```

```
     * Used to indicate whether or not an event can have its default action
     * prevented. If the default action can be prevented the value is true,
     * else the value is false.
     */
    public boolean getCancelable();

    /**
     *  Used to specify the time (in milliseconds relative to the epoch) at
     * which the event was created. Due to the fact that some systems may
     * not provide this information the value of <code>timeStamp</code> may
     * be not available for all events. When not available, a value of 0
     * will be returned. Examples of epoch time are the time of the system
     * start or 0:0:0 UTC 1st January 1970.
     */
    public long getTimeStamp();

    /**
     * The <code>stopPropagation</code> method is used prevent further
     * propagation of an event during event flow. If this method is called
     * by any <code>EventListener</code> the event will cease propagating
     * through the tree. The event will complete dispatch to all listeners
     * on the current <code>EventTarget</code> before event flow stops. This
     * method may be used during any stage of event flow.
     */
    public void stopPropagation();

    /**
     * If an event is cancelable, the <code>preventDefault</code> method is
     * used to signify that the event is to be canceled, meaning any default
     * action normally taken by the implementation as a result of the event
     * will not occur. If, during any stage of event flow, the
     * <code>preventDefault</code> method is called the event is canceled.
     * Any default action associated with the event will not occur. Calling
     * this method for a non-cancelable event has no effect. Once
     * <code>preventDefault</code> has been called it will remain in effect
     * throughout the remainder of the event's propagation. This method may
     * be used during any stage of event flow.
     */
    public void preventDefault();

    /**
     * The <code>initEvent</code> method is used to initialize the value of an
     * <code>Event</code> created through the <code>DocumentEvent</code>
     * interface. This method may only be called before the
     * <code>Event</code> has been dispatched via the
     * <code>dispatchEvent</code> method, though it may be called multiple
     * times during that phase if necessary. If called multiple times the
     * final invocation takes precedence. If called from a subclass of
     * <code>Event</code> interface only the values specified in the
     * <code>initEvent</code> method are modified, all other attributes are
     * left unchanged.
     * @param eventTypeArg Specifies the event type. This type may be any
     *   event type currently defined in this specification or a new event
```

```
     *   type.. The string must be an XML name. Any new event type must not
     *   begin with any upper, lower, or mixed case version of the string
     *   "DOM". This prefix is reserved for future DOM event sets. It is
     *   also strongly recommended that third parties adding their own
     *   events use their own prefix to avoid confusion and lessen the
     *   probability of conflicts with other new events.
     * @param canBubbleArg Specifies whether or not the event can bubble.
     * @param cancelableArg Specifies whether or not the event's default
     *   action can be prevented.
     */
    public void initEvent(String eventTypeArg,
                          boolean canBubbleArg,
                          boolean cancelableArg);

}
```

39.3 EventException.java

Secuencia 385: org/w3c/dom/events/EventException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
```

```
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

/**
 *  Event operations may throw an <code>EventException</code> as specified in
 * their method descriptions.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public class EventException extends RuntimeException {
    public EventException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
    // EventExceptionCode
    /**
     *  If the <code>Event</code>'s type was not specified by initializing the
     * event before the method was called. Specification of the Event's type
     * as <code>null</code> or an empty string will also trigger this
     * exception.
     */
    public static final short UNSPECIFIED_EVENT_TYPE_ERR = 0;

}
```

39.4 EventListener.java

Secuencia 386: org/w3c/dom/events/EventListener.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

/**
 *  The <code>EventListener</code> interface is the primary method for
 * handling events. Users implement the <code>EventListener</code> interface
 * and register their listener on an <code>EventTarget</code> using the
 * <code>AddEventListener</code> method. The users should also remove their
 * <code>EventListener</code> from its <code>EventTarget</code> after they
 * have completed using the listener.
 * <p> When a <code>Node</code> is copied using the <code>cloneNode</code>
 * method the <code>EventListener</code>s attached to the source
 * <code>Node</code> are not attached to the copied <code>Node</code>. If
 * the user wishes the same <code>EventListener</code>s to be added to the
 * newly created copy the user must add them manually.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface EventListener {
    /**
     *  This method is called whenever an event occurs of the type for which
     * the <code> EventListener</code> interface was registered.
     * @param evt  The <code>Event</code> contains contextual information
     *   about the event. It also contains the <code>stopPropagation</code>
     *   and <code>preventDefault</code> methods which are used in
     *   determining the event's flow and default action.
     */
    public void handleEvent(Event evt);

```

```
}
```

39.5 EventTarget.java

Secuencia 387: org/w3c/dom/events/EventTarget.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

/**
 *  The <code>EventTarget</code> interface is implemented by all
 * <code>Nodes</code> in an implementation which supports the DOM Event
 * Model. Therefore, this interface can be obtained by using
 * binding-specific casting methods on an instance of the <code>Node</code>
```

```
 * interface. The interface allows registration and removal of
 * <code>EventListeners</code> on an <code>EventTarget</code> and dispatch
 * of events to that <code>EventTarget</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface EventTarget {
    /**
     * This method allows the registration of event listeners on the event
     * target. If an <code>EventListener</code> is added to an
     * <code>EventTarget</code> while it is processing an event, it will not
     * be triggered by the current actions but may be triggered during a
     * later stage of event flow, such as the bubbling phase.
     * <br> If multiple identical <code>EventListener</code>s are registered
     * on the same <code>EventTarget</code> with the same parameters the
     * duplicate instances are discarded. They do not cause the
     * <code>EventListener</code> to be called twice and since they are
     * discarded they do not need to be removed with the
     * <code>removeEventListener</code> method.
     * @param type The event type for which the user is registering
     * @param listener The <code>listener</code> parameter takes an interface
     *   implemented by the user which contains the methods to be called
     *   when the event occurs.
     * @param useCapture If true, <code>useCapture</code> indicates that the
     *   user wishes to initiate capture. After initiating capture, all
     *   events of the specified type will be dispatched to the registered
     *   <code>EventListener</code> before being dispatched to any
     *   <code>EventTargets</code> beneath them in the tree. Events which
     *   are bubbling upward through the tree will not trigger an
     *   <code>EventListener</code> designated to use capture.
     */
    public void addEventListener(String type,
                                 EventListener listener,
                                 boolean useCapture);

    /**
     * This method allows the removal of event listeners from the event
     * target. If an <code>EventListener</code> is removed from an
     * <code>EventTarget</code> while it is processing an event, it will not
     * be triggered by the current actions. <code>EventListener</code>s can
     * never be invoked after being removed.
     * <br>Calling <code>removeEventListener</code> with arguments which do
     * not identify any currently registered <code>EventListener</code> on
     * the <code>EventTarget</code> has no effect.
     * @param type Specifies the event type of the <code>EventListener</code>
     *   being removed.
     * @param listener The <code>EventListener</code> parameter indicates the
     *   <code>EventListener </code> to be removed.
     * @param useCapture Specifies whether the <code>EventListener</code>
     *   being removed was registered as a capturing listener or not. If a
     *   listener was registered twice, one with capture and one without,
     *   each must be removed separately. Removal of a capturing listener
```

```
     *   does not affect a non-capturing version of the same listener, and
     *   vice versa.
     */
    public void removeEventListener(String type,
                                    EventListener listener,
                                    boolean useCapture);

    /**
     * This method allows the dispatch of events into the implementations
     * event model. Events dispatched in this manner will have the same
     * capturing and bubbling behavior as events dispatched directly by the
     * implementation. The target of the event is the
     * <code> EventTarget</code> on which <code>dispatchEvent</code> is
     * called.
     * @param evt Specifies the event type, behavior, and contextual
     *   information to be used in processing the event.
     * @return The return value of <code>dispatchEvent</code> indicates
     *   whether any of the listeners which handled the event called
     *   <code>preventDefault</code>. If <code>preventDefault</code> was
     *   called the value is false, else the value is true.
     * @exception EventException
     *   UNSPECIFIED_EVENT_TYPE_ERR: Raised if the <code>Event</code>'s type
     *   was not specified by initializing the event before
     *   <code>dispatchEvent</code> was called. Specification of the
     *   <code>Event</code>'s type as <code>null</code> or an empty string
     *   will also trigger this exception.
     */
    public boolean dispatchEvent(Event evt)
                                 throws EventException;

}
```

39.6 MouseEvent.java

Secuencia 388: org/w3c/dom/events/MouseEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

import org.w3c.dom.views.AbstractView;

/**
 * The <code>MouseEvent</code> interface provides specific contextual
 * information associated with Mouse events.
 * <p>The <code>detail</code> attribute inherited from <code>UIEvent</code>
 * indicates the number of times a mouse button has been pressed and
 * released over the same screen location during a user action. The
 * attribute value is 1 when the user begins this action and increments by 1
 * for each full sequence of pressing and releasing. If the user moves the
 * mouse between the mousedown and mouseup the value will be set to 0,
 * indicating that no click is occurring.
 * <p>In the case of nested elements mouse events are always targeted at the
 * most deeply nested element. Ancestors of the targeted element may use
 * bubbling to obtain notification of mouse events which occur within its
 * descendent elements.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface MouseEvent extends UIEvent {
    /**
     * The horizontal coordinate at which the event occurred relative to the
     * origin of the screen coordinate system.
     */
    public int getScreenX();

    /**
```

```
     * The vertical coordinate at which the event occurred relative to the
     * origin of the screen coordinate system.
     */
    public int getScreenY();

    /**
     * The horizontal coordinate at which the event occurred relative to the
     * DOM implementation's client area.
     */
    public int getClientX();

    /**
     * The vertical coordinate at which the event occurred relative to the DOM
     * implementation's client area.
     */
    public int getClientY();

    /**
     * Used to indicate whether the 'ctrl' key was depressed during the firing
     * of the event.
     */
    public boolean getCtrlKey();

    /**
     * Used to indicate whether the 'shift' key was depressed during the
     * firing of the event.
     */
    public boolean getShiftKey();

    /**
     * Used to indicate whether the 'alt' key was depressed during the firing
     * of the event. On some platforms this key may map to an alternative
     * key name.
     */
    public boolean getAltKey();

    /**
     * Used to indicate whether the 'meta' key was depressed during the firing
     * of the event. On some platforms this key may map to an alternative
     * key name.
     */
    public boolean getMetaKey();

    /**
     * During mouse events caused by the depression or release of a mouse
     * button, <code>button</code> is used to indicate which mouse button
     * changed state. The values for <code>button</code> range from zero to
     * indicate the left button of the mouse, one to indicate the middle
     * button if present, and two to indicate the right button. For mice
     * configured for left handed use in which the button actions are
     * reversed the values are instead read from right to left.
     */
    public short getButton();
```

```

    /**
     * Used to identify a secondary <code>EventTarget</code> related to a UI
     * event. Currently this attribute is used with the mouseover event to
     * indicate the <code>EventTarget</code> which the pointing device
     * exited and with the mouseout event to indicate the
     * <code>EventTarget</code> which the pointing device entered.
     */
    public EventTarget getRelatedTarget();

    /**
     * The <code>initMouseEvent</code> method is used to initialize the value
     * of a <code>MouseEvent</code> created through the
     * <code>DocumentEvent</code> interface. This method may only be called
     * before the <code>MouseEvent</code> has been dispatched via the
     * <code>dispatchEvent</code> method, though it may be called multiple
     * times during that phase if necessary. If called multiple times, the
     * final invocation takes precedence.
     * @param typeArg Specifies the event type.
     * @param canBubbleArg Specifies whether or not the event can bubble.
     * @param cancelableArg Specifies whether or not the event's default
     *   action can be prevented.
     * @param viewArg Specifies the <code>Event</code>'s
     *   <code>AbstractView</code>.
     * @param detailArg Specifies the <code>Event</code>'s mouse click count.
     * @param screenXArg Specifies the <code>Event</code>'s screen x
     *   coordinate
     * @param screenYArg Specifies the <code>Event</code>'s screen y
     *   coordinate
     * @param clientXArg Specifies the <code>Event</code>'s client x
     *   coordinate
     * @param clientYArg Specifies the <code>Event</code>'s client y
     *   coordinate
     * @param ctrlKeyArg Specifies whether or not control key was depressed
     *   during the <code>Event</code>.
     * @param altKeyArg Specifies whether or not alt key was depressed during
     *   the <code>Event</code>.
     * @param shiftKeyArg Specifies whether or not shift key was depressed
     *   during the <code>Event</code>.
     * @param metaKeyArg Specifies whether or not meta key was depressed
     *   during the <code>Event</code>.
     * @param buttonArg Specifies the <code>Event</code>'s mouse button.
     * @param relatedTargetArg Specifies the <code>Event</code>'s related
     *   <code>EventTarget</code>.
     */
    public void initMouseEvent(String typeArg,
                               boolean canBubbleArg,
                               boolean cancelableArg,
                               AbstractView viewArg,
                               int detailArg,
                               int screenXArg,
                               int screenYArg,
                               int clientXArg,
```

```
                                int clientYArg,
                                boolean ctrlKeyArg,
                                boolean altKeyArg,
                                boolean shiftKeyArg,
                                boolean metaKeyArg,
                                short buttonArg,
                                EventTarget relatedTargetArg);

}
```

39.7 MutationEvent.java

Secuencia 389: org/w3c/dom/events/MutationEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */
```

```java

package org.w3c.dom.events;

import org.w3c.dom.Node;

/**
 * The <code>MutationEvent</code> interface provides specific contextual
 * information associated with Mutation events.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface MutationEvent extends Event {
    // attrChangeType
    /**
     * The <code>Attr</code> was modified in place.
     */
    public static final short MODIFICATION              = 1;
    /**
     * The <code>Attr</code> was just added.
     */
    public static final short ADDITION                  = 2;
    /**
     * The <code>Attr</code> was just removed.
     */
    public static final short REMOVAL                   = 3;

    /**
     *  <code>relatedNode</code> is used to identify a secondary node related
     * to a mutation event. For example, if a mutation event is dispatched
     * to a node indicating that its parent has changed, the
     * <code>relatedNode</code> is the changed parent. If an event is
     * instead dispatched to a subtree indicating a node was changed within
     * it, the <code>relatedNode</code> is the changed node. In the case of
     * the DOMAttrModified event it indicates the <code>Attr</code> node
     * which was modified, added, or removed.
     */
    public Node getRelatedNode();

    /**
     *  <code>prevValue</code> indicates the previous value of the
     * <code>Attr</code> node in DOMAttrModified events, and of the
     * <code>CharacterData</code> node in DOMCharacterDataModified events.
     */
    public String getPrevValue();

    /**
     *  <code>newValue</code> indicates the new value of the <code>Attr</code>
     * node in DOMAttrModified events, and of the <code>CharacterData</code>
     * node in DOMCharacterDataModified events.
     */
    public String getNewValue();

```

```
    /**
     *  <code>attrName</code> indicates the name of the changed
     * <code>Attr</code> node in a DOMAttrModified event.
     */
    public String getAttrName();

    /**
     *  <code>attrChange</code> indicates the type of change which triggered
     * the DOMAttrModified event. The values can be <code>MODIFICATION</code>
     * , <code>ADDITION</code>, or <code>REMOVAL</code>.
     */
    public short getAttrChange();

    /**
     * The <code>initMutationEvent</code> method is used to initialize the
     * value of a <code>MutationEvent</code> created through the
     * <code>DocumentEvent</code> interface. This method may only be called
     * before the <code>MutationEvent</code> has been dispatched via the
     * <code>dispatchEvent</code> method, though it may be called multiple
     * times during that phase if necessary. If called multiple times, the
     * final invocation takes precedence.
     * @param typeArg Specifies the event type.
     * @param canBubbleArg Specifies whether or not the event can bubble.
     * @param cancelableArg Specifies whether or not the event's default
     *   action can be prevented.
     * @param relatedNodeArg Specifies the <code>Event</code>'s related Node.
     * @param prevValueArg Specifies the <code>Event</code>'s
     *   <code>prevValue</code> attribute. This value may be null.
     * @param newValueArg Specifies the <code>Event</code>'s
     *   <code>newValue</code> attribute. This value may be null.
     * @param attrNameArg Specifies the <code>Event</code>'s
     *   <code>attrName</code> attribute. This value may be null.
     * @param attrChangeArg Specifies the <code>Event</code>'s
     *   <code>attrChange</code> attribute
     */
    public void initMutationEvent(String typeArg,
                                  boolean canBubbleArg,
                                  boolean cancelableArg,
                                  Node relatedNodeArg,
                                  String prevValueArg,
                                  String newValueArg,
                                  String attrNameArg,
                                  short attrChangeArg);

}
```

39.8 UIEvent.java

Secuencia 390: org/w3c/dom/events/UIEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.events;

import org.w3c.dom.views.AbstractView;

/**
 * The <code>UIEvent</code> interface provides specific contextual information
 * associated with User Interface events.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113'>Document
     Object Model (DOM) Level 2 Events Specification</a>.
 * @since DOM Level 2
 */
public interface UIEvent extends Event {
    /**
     * The <code>view</code> attribute identifies the <code>AbstractView</code>
     *  from which the event was generated.
     */
```

```
    public AbstractView getView();

    /**
     * Specifies some detail information about the <code>Event</code>,
     * depending on the type of event.
     */
    public int getDetail();

    /**
     * The <code>initUIEvent</code> method is used to initialize the value of
     * a <code>UIEvent</code> created through the <code>DocumentEvent</code>
     * interface. This method may only be called before the
     * <code>UIEvent</code> has been dispatched via the
     * <code>dispatchEvent</code> method, though it may be called multiple
     * times during that phase if necessary. If called multiple times, the
     * final invocation takes precedence.
     * @param typeArg Specifies the event type.
     * @param canBubbleArg Specifies whether or not the event can bubble.
     * @param cancelableArg Specifies whether or not the event's default
     *   action can be prevented.
     * @param viewArg Specifies the <code>Event</code>'s
     *   <code>AbstractView</code>.
     * @param detailArg Specifies the <code>Event</code>'s detail.
     */
    public void initUIEvent(String typeArg,
                            boolean canBubbleArg,
                            boolean cancelableArg,
                            AbstractView viewArg,
                            int detailArg);

}
```

40 Dom Html (org.w3c.dom.html package)

40.1 HTMLAnchorElement.java

Secuencia 391: org/w3c/dom/html/HTMLAnchorElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  The anchor element. See the  A element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLAnchorElement extends HTMLElement {
    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  The character encoding of the linked resource. See the  charset
     * attribute definition in HTML 4.0.
     */
    public String getCharset();
    public void setCharset(String charset);

    /**
     *  Comma-separated list of lengths, defining an active region geometry.
     * See also <code>shape</code> for the shape of the region. See the
     * coords attribute definition in HTML 4.0.
     */
    public String getCoords();
```

```
    public void setCoords(String coords);

    /**
     *  The URI of the linked resource. See the  href attribute definition in
     * HTML 4.0.
     */
    public String getHref();
    public void setHref(String href);

    /**
     *  Language code of the linked resource. See the  hreflang attribute
     * definition in HTML 4.0.
     */
    public String getHreflang();
    public void setHreflang(String hreflang);

    /**
     *  Anchor name. See the  name attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Forward link type. See the  rel attribute definition in HTML 4.0.
     */
    public String getRel();
    public void setRel(String rel);

    /**
     *  Reverse link type. See the  rev attribute definition in HTML 4.0.
     */
    public String getRev();
    public void setRev(String rev);

    /**
     *  The shape of the active area. The coordinates are given by
     * <code>coords</code> . See the  shape attribute definition in HTML 4.0.
     */
    public String getShape();
    public void setShape(String shape);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  Frame to render the resource in. See the  target attribute definition
     * in HTML 4.0.
     */
    public String getTarget();
```

```
    public void setTarget(String target);

    /**
     *  Advisory content type. See the  type attribute definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

    /**
     *  Removes keyboard focus from this element.
     */
    public void blur();

    /**
     *  Gives keyboard focus to this element.
     */
    public void focus();

}
```

40.2 HTMLAppletElement.java

Secuencia 392: org/w3c/dom/html/HTMLAppletElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
```

```
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  An embedded Java applet. See the  APPLET element definition in HTML 4.0.
 * This element is deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLAppletElement extends HTMLElement {
    /**
     *  Aligns this object (vertically or horizontally)  with respect to its
     * surrounding text. See the  align attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Alternate text for user agents not rendering the normal content of
     * this element. See the  alt attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getAlt();
    public void setAlt(String alt);

    /**
     *  Comma-separated archive list. See the  archive attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getArchive();
    public void setArchive(String archive);

    /**
     *  Applet class file.  See the  code attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getCode();
    public void setCode(String code);

    /**
     *  Optional base URI for applet. See the  codebase attribute definition
     * in HTML 4.0. This attribute is deprecated in HTML 4.0.
```

```
     */
    public String getCodeBase();
    public void setCodeBase(String codeBase);

    /**
     *  Override height. See the  height attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getHeight();
    public void setHeight(String height);

    /**
     *  Horizontal space to the left and right of this image, applet, or
     * object. See the  hspace attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getHspace();
    public void setHspace(String hspace);

    /**
     *  The name of the applet. See the  name attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Serialized applet file. See the  object attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getObject();
    public void setObject(String object);

    /**
     *  Vertical space above and below this image, applet, or object. See the
     * vspace attribute definition in HTML 4.0. This attribute is deprecated
     * in HTML 4.0.
     */
    public String getVspace();
    public void setVspace(String vspace);

    /**
     *  Override width. See the  width attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

}
```

40.3 HTMLAreaElement.java

Secuencia 393: org/w3c/dom/html/HTMLAreaElement.java

```java
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Client-side image map area definition. See the  AREA element definition in
 * HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLAreaElement extends HTMLElement {
    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
```

```
     */
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  Alternate text for user agents not rendering the normal content of
     * this element. See the  alt attribute definition in HTML 4.0.
     */
    public String getAlt();
    public void setAlt(String alt);

    /**
     *  Comma-separated list of lengths, defining an active region geometry.
     * See also <code>shape</code> for the shape of the region. See the
     * coords attribute definition in HTML 4.0.
     */
    public String getCoords();
    public void setCoords(String coords);

    /**
     *  The URI of the linked resource. See the  href attribute definition in
     * HTML 4.0.
     */
    public String getHref();
    public void setHref(String href);

    /**
     *  Specifies that this area is inactive, i.e., has no associated action.
     * See the  nohref attribute definition in HTML 4.0.
     */
    public boolean getNoHref();
    public void setNoHref(boolean noHref);

    /**
     *  The shape of the active area. The coordinates are given by
     * <code>coords</code> . See the  shape attribute definition in HTML 4.0.
     */
    public String getShape();
    public void setShape(String shape);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  Frame to render the resource in. See the  target attribute definition
     * in HTML 4.0.
     */
    public String getTarget();
    public void setTarget(String target);
```

```

}
```

40.4 HTMLBRElement.java

Secuencia 394: org/w3c/dom/html/HTMLBRElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Force a line break. See the  BR element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
```

```
 */
public interface HTMLBRElement extends HTMLElement {
    /**
     *  Control flow of text around floats. See the  clear attribute definition
     *  in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getClear();
    public void setClear(String clear);

}
```

40.5 HTMLBaseElement.java

Secuencia 395: org/w3c/dom/html/HTMLBaseElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
```

```
 */

package org.w3c.dom.html;

/**
 *  Document base URI. See the  BASE element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLBaseElement extends HTMLElement {
    /**
     *  The base URI. See the  href attribute definition in HTML 4.0.
     */
    public String getHref();
    public void setHref(String href);

    /**
     *  The default target frame. See the  target attribute definition in HTML
     * 4.0.
     */
    public String getTarget();
    public void setTarget(String target);

}
```

40.6 HTMLBaseFontElement.java

Secuencia 396: org/w3c/dom/html/HTMLBaseFontElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
```

```
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Base font. See the  BASEFONT element definition in HTML 4.0. This element
 * is deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLBaseFontElement extends HTMLElement {
    /**
     *  Font color. See the  color attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getColor();
    public void setColor(String color);

    /**
     *  Font face identifier. See the  face attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getFace();
    public void setFace(String face);

    /**
     *  Font size. See the  size attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getSize();
    public void setSize(String size);

}
```

40.7 HTMLBodyElement.java

Secuencia 397: org/w3c/dom/html/HTMLBodyElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  The HTML document body. This element is always present in the DOM API,
 * even if the tags are not present in the source document. See the  BODY
 * element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLBodyElement extends HTMLElement {
    /**
     *  Color of active links (after mouse-button down, but before
     * mouse-button up). See the  alink attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
```

```
     */
    public String getALink();
    public void setALink(String aLink);

    /**
     *  URI of the background texture tile image. See the  background
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getBackground();
    public void setBackground(String background);

    /**
     *  Document background color. See the  bgcolor attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBgColor();
    public void setBgColor(String bgColor);

    /**
     *  Color of links that are not active and unvisited. See the  link
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getLink();
    public void setLink(String link);

    /**
     *  Document text color. See the  text attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getText();
    public void setText(String text);

    /**
     *  Color of links that have been visited by the user. See the  vlink
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getVLink();
    public void setVLink(String vLink);

}
```

40.8 HTMLButtonElement.java

Secuencia 398: org/w3c/dom/html/HTMLButtonElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Push button. See the  BUTTON element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLButtonElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
```

```
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  Form control or object name when submitted with a form. See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  The type of button. See the  type attribute definition in HTML 4.0.
     */
    public String getType();

    /**
     *  The current form control value. See the  value attribute definition in
     * HTML 4.0.
     */
    public String getValue();
    public void setValue(String value);

}
```

40.9 HTMLCollection.java

Secuencia 399: org/w3c/dom/html/HTMLCollection.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.Node;

/**
 *  An <code>HTMLCollection</code> is a list of nodes. An individual node may
 * be accessed by either ordinal index or the node's<code>name</code> or
 * <code>id</code> attributes.  Note: Collections in the HTML DOM are assumed
 * to be  live meaning that they are automatically updated when the
 * underlying document is changed.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLCollection {
    /**
     *  This attribute specifies the length or  size of the list.
     */
    public int getLength();

    /**
     *  This method retrieves a node specified by ordinal index. Nodes are
     * numbered in tree order (depth-first traversal order).
     * @param index  The index of the node to be fetched. The index origin is
     *   0.
```

```
     * @return  The <code>Node</code> at the corresponding position upon
     *   success. A value of <code>null</code> is returned if the index is
     *   out of range.
     */
    public Node item(int index);

    /**
     *  This method retrieves a <code>Node</code> using a name. It first
     * searches for a <code>Node</code> with a matching <code>id</code>
     * attribute. If it doesn't find one, it then searches for a
     * <code>Node</code> with a matching <code>name</code> attribute, but
     * only on those elements that are allowed a name attribute.
     * @param name  The name of the <code>Node</code> to be fetched.
     * @return  The <code>Node</code> with a <code>name</code> or
     *   <code>id</code> attribute whose value corresponds to the specified
     *   string. Upon failure (e.g., no node with this name exists), returns
     *   <code>null</code> .
     */
    public Node namedItem(String name);

}
```

40.10 HTMLDListElement.java

Secuencia 400: org/w3c/dom/html/HTMLDListElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Definition list. See the  DL element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLDListElement extends HTMLElement {
    /**
     *  Reduce spacing between list items. See the  compact attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

40.11 HTMLDOMImplementation.java

Secuencia 401: `org/w3c/dom/html/HTMLDOMImplementation.java`

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.DOMImplementation;

/**
 *  The <code>HTMLDOMImplementation</code> interface extends the
 * <code>DOMImplementation</code> interface with a method for creating an
 * HTML document instance.
 * @since DOM Level 2
 */
public interface HTMLDOMImplementation extends DOMImplementation {
    /**
     *  Creates an <code>HTMLDocument</code> object with the minimal tree made
     * of the following elements: <code>HTML</code> , <code>HEAD</code> ,
     * <code>TITLE</code> , and <code>BODY</code> .
     * @param title  The title of the document to be set as the content of the
     *   <code>TITLE</code> element, through a child <code>Text</code> node.
     * @return  A new <code>HTMLDocument</code> object.
     */
    public HTMLDocument createHTMLDocument(String title);

}
```

40.12 HTMLDirectoryElement.java

Secuencia 402: org/w3c/dom/html/HTMLDirectoryElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Directory list. See the  DIR element definition in HTML 4.0. This element
 * is deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLDirectoryElement extends HTMLElement {
    /**
     *  Reduce spacing between list items. See the  compact attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

40.13 HTMLDivElement.java

Secuencia 403: org/w3c/dom/html/HTMLDivElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Generic block container. See the  DIV element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLDivElement extends HTMLElement {
    /**
```

```
     *  Horizontal text alignment. See the  align attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

}
```

40.14 HTMLDocument.java

Secuencia 404: org/w3c/dom/html/HTMLDocument.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;
```

```

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

/**
 *  An <code>HTMLDocument</code> is the root of the HTML hierarchy and holds
 * the entire content. Besides providing access to the hierarchy, it also
 * provides some convenience methods for accessing certain sets of
 * information from the document.
 * <p> The following properties have been deprecated in favor of the
 * corresponding ones for the <code>BODY</code> element: alinkColor background
 *  bgColor fgColor linkColor vlinkColor In DOM Level 2, the method
 * <code>getElementById</code> is inherited from the <code>Document</code>
 * interface where it was moved.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLDocument extends Document {
    /**
     *  The title of a document as specified by the <code>TITLE</code> element
     * in the head of the document.
     */
    public String getTitle();
    public void setTitle(String title);

    /**
     *  Returns the URI  of the page that linked to this page. The value is an
     * empty string if the user navigated to the page directly (not through a
     * link, but, for example, via a bookmark).
     */
    public String getReferrer();

    /**
     *  The domain name of the server that served the document, or
     * <code>null</code> if the server cannot be identified by a domain name.
     */
    public String getDomain();

    /**
     *  The complete URI  of the document.
     */
    public String getURL();

    /**
     *  The element that contains the content for the document. In documents
     * with <code>BODY</code> contents, returns the <code>BODY</code>
     * element. In frameset documents, this returns the outermost
     * <code>FRAMESET</code> element.
     */
    public HTMLElement getBody();
    public void setBody(HTMLElement body);

    /**
```

```
     *  A collection of all the <code>IMG</code> elements in a document. The
     * behavior is limited to <code>IMG</code> elements for backwards
     * compatibility.
     */
    public HTMLCollection getImages();

    /**
     *  A collection of all the <code>OBJECT</code> elements that include
     * applets and <code>APPLET</code> ( deprecated ) elements in a document.
     */
    public HTMLCollection getApplets();

    /**
     *  A collection of all <code>AREA</code> elements and anchor (
     * <code>A</code> ) elements in a document with a value for the
     * <code>href</code> attribute.
     */
    public HTMLCollection getLinks();

    /**
     *  A collection of all the forms of a document.
     */
    public HTMLCollection getForms();

    /**
     *  A collection of all the anchor (<code>A</code> ) elements in a document
     *  with a value for the <code>name</code> attribute. Note. For reasons
     * of backwards compatibility, the returned set of anchors only contains
     * those anchors created with the <code>name</code>  attribute, not those
     * created with the <code>id</code> attribute.
     */
    public HTMLCollection getAnchors();

    /**
     *  The cookies associated with this document. If there are none, the
     * value is an empty string. Otherwise, the value is a string: a
     * semicolon-delimited list of "name, value" pairs for all the cookies
     * associated with the page. For example,
     * <code>name=value;expires=date</code> .
     */
    public String getCookie();
    public void setCookie(String cookie);

    /**
     *  Note. This method and the ones following  allow a user to add to or
     * replace the structure model of a document using strings of unparsed
     * HTML. At the time of  writing alternate methods for providing similar
     * functionality for  both HTML and XML documents were being considered.
     * The following methods may be deprecated at some point in the future in
     * favor of a more general-purpose mechanism.
     * <br> Open a document stream for writing. If a document exists in the
     * target, this method clears it.
     */
```

```
    public void open();

    /**
     *  Closes a document stream opened by <code>open()</code> and forces
     * rendering.
     */
    public void close();

    /**
     *  Write a string of text to a document stream opened by
     * <code>open()</code> . The text is parsed into the document's structure
     * model.
     * @param text  The string to be parsed into some structure in the
     *   document structure model.
     */
    public void write(String text);

    /**
     *  Write a string of text followed by a newline character to a document
     * stream opened by <code>open()</code> . The text is parsed into the
     * document's structure model.
     * @param text  The string to be parsed into some structure in the
     *   document structure model.
     */
    public void writeln(String text);

    /**
     *  Returns the (possibly empty) collection of elements whose
     * <code>name</code> value is given by <code>elementName</code> .
     * @param elementName  The <code>name</code> attribute value for an
     *   element.
     * @return  The matching elements.
     */
    public NodeList getElementsByName(String elementName);

}
```

40.15 HTMLElement.java

Secuencia 405: org/w3c/dom/html/HTMLElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.Element;

/**
 *  All HTML element interfaces derive from this class. Elements that only
 * expose the HTML core attributes are represented by the base
 * <code>HTMLElement</code> interface. These elements are as follows:  HEAD
 * special: SUB, SUP, SPAN, BDO font: TT, I, B, U, S, STRIKE, BIG, SMALL
 * phrase: EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE, ACRONYM, ABBR list:
 * DD, DT NOFRAMES, NOSCRIPT ADDRESS, CENTER The <code>style</code> attribute
 * of an HTML element is accessible through the
 * <code>ElementCSSInlineStyle</code> interface which is defined in the  .
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLElement extends Element {
    /**
     *  The element's identifier. See the  id attribute definition in HTML 4.0.
     */
    public String getId();
    public void setId(String id);

    /**
     *  The element's advisory title. See the  title attribute definition in
```

```
     * HTML 4.0.
     */
    public String getTitle();
    public void setTitle(String title);

    /**
     *  Language code defined in RFC 1766. See the  lang attribute definition
     * in HTML 4.0.
     */
    public String getLang();
    public void setLang(String lang);

    /**
     *  Specifies the base direction of directionally neutral text and the
     * directionality of tables. See the  dir attribute definition in HTML
     * 4.0.
     */
    public String getDir();
    public void setDir(String dir);

    /**
     *  The class attribute of the element. This attribute has been renamed
     * due to conflicts with the "class" keyword exposed by many languages.
     * See the  class attribute definition in HTML 4.0.
     */
    public String getClassName();
    public void setClassName(String className);

}
```

40.16 HTMLFieldSetElement.java

Secuencia 406: `org/w3c/dom/html/HTMLFieldSetElement.java`

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Organizes form controls into logical groups. See the   FIELDSET  element
 * definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLFieldSetElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

}
```

40.17 HTMLFontElement.java

Secuencia 407: org/w3c/dom/html/HTMLFontElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Local change to font. See the  FONT element definition in HTML 4.0. This
 * element is deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLFontElement extends HTMLElement {
    /**
     *  Font color. See the  color attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getColor();
    public void setColor(String color);

    /**
     *  Font face identifier. See the  face attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getFace();
    public void setFace(String face);

    /**
```

```
     *  Font size. See the  size attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getSize();
    public void setSize(String size);

}
```

40.18 HTMLFormElement.java

Secuencia 408: org/w3c/dom/html/HTMLFormElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;
```

```

/**
 *  The <code>FORM</code> element encompasses behavior similar to a collection
 * and an element. It provides direct access to the contained input elements
 * as well as the attributes of the form element. See the  FORM element
 * definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLFormElement extends HTMLElement {
    /**
     *  Returns a collection of all control elements in the form.
     */
    public HTMLCollection getElements();

    /**
     *  The number of form controls in the form.
     */
    public int getLength();

    /**
     *  Names the form.
     */
    public String getName();
    public void setName(String name);

    /**
     *  List of character sets supported by the server. See the
     * accept-charset attribute definition in HTML 4.0.
     */
    public String getAcceptCharset();
    public void setAcceptCharset(String acceptCharset);

    /**
     *  Server-side form handler. See the  action attribute definition in HTML
     * 4.0.
     */
    public String getAction();
    public void setAction(String action);

    /**
     *  The content type of the submitted form,  generally
     * "application/x-www-form-urlencoded".  See the  enctype attribute
     * definition in HTML 4.0.
     */
    public String getEnctype();
    public void setEnctype(String enctype);

    /**
     *  HTTP method used to submit form. See the  method attribute definition
     * in HTML 4.0.
     */
    public String getMethod();
```

```
    public void setMethod(String method);

    /**
     *  Frame to render the resource in. See the  target attribute definition
     * in HTML 4.0.
     */
    public String getTarget();
    public void setTarget(String target);

    /**
     *  Submits the form. It performs the same action as a  submit button.
     */
    public void submit();

    /**
     *  Restores a form element's default values. It performs  the same action
     * as a reset button.
     */
    public void reset();

}
```

40.19 HTMLFrameElement.java

Secuencia 409: org/w3c/dom/html/HTMLFrameElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.Document;

/**
 *  Create a frame. See the  FRAME element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLFrameElement extends HTMLElement {
    /**
     *  Request frame borders. See the  frameborder attribute definition in
     * HTML 4.0.
     */
    public String getFrameBorder();
    public void setFrameBorder(String frameBorder);

    /**
     *  URI designating a long description of this image or frame. See the
     * longdesc attribute definition in HTML 4.0.
     */
    public String getLongDesc();
    public void setLongDesc(String longDesc);

    /**
     *  Frame margin height, in pixels. See the  marginheight attribute
     * definition in HTML 4.0.
     */
    public String getMarginHeight();
    public void setMarginHeight(String marginHeight);

    /**
     *  Frame margin width, in pixels. See the  marginwidth attribute
     * definition in HTML 4.0.
     */
    public String getMarginWidth();
    public void setMarginWidth(String marginWidth);

    /**
     *  The frame name (object of the <code>target</code> attribute). See the
```

```
     * name attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  When true, forbid user from resizing frame. See the  noresize
     * attribute definition in HTML 4.0.
     */
    public boolean getNoResize();
    public void setNoResize(boolean noResize);

    /**
     *  Specify whether or not the frame should have scrollbars. See the
     * scrolling attribute definition in HTML 4.0.
     */
    public String getScrolling();
    public void setScrolling(String scrolling);

    /**
     *  A URI designating the initial frame contents. See the  src attribute
     * definition in HTML 4.0.
     */
    public String getSrc();
    public void setSrc(String src);

    /**
     *  The document this frame contains, if there is any and it is available,
     * or <code>null</code> otherwise.
     * @since DOM Level 2
     */
    public Document getContentDocument();

}
```

40.20 HTMLFrameSetElement.java

Secuencia 410: org/w3c/dom/html/HTMLFrameSetElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Create a grid of frames. See the  FRAMESET element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLFrameSetElement extends HTMLElement {
    /**
     *  The number of columns of frames in the frameset. See the  cols
     * attribute definition in HTML 4.0.
     */
    public String getCols();
    public void setCols(String cols);

    /**
     *  The number of rows of frames in the frameset. See the  rows attribute
     * definition in HTML 4.0.
     */
    public String getRows();
    public void setRows(String rows);

}
```

40.21 HTMLHRElement.java

Secuencia 411: org/w3c/dom/html/HTMLHRElement.java

```java
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Create a horizontal rule. See the  HR element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLHRElement extends HTMLElement {
    /**
     *  Align the rule on the page. See the  align attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
```

```
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Indicates to the user agent that there should be no shading in the
     * rendering of this element. See the  noshade attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getNoShade();
    public void setNoShade(boolean noShade);

    /**
     *  The height of the rule. See the  size attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getSize();
    public void setSize(String size);

    /**
     *  The width of the rule. See the  width attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

}
```

40.22 HTMLHeadElement.java

Secuencia 412: org/w3c/dom/html/HTMLHeadElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Document head information. See the  HEAD element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLHeadElement extends HTMLElement {
    /**
     *  URI designating a metadata profile. See the  profile attribute
     * definition in HTML 4.0.
     */
    public String getProfile();
    public void setProfile(String profile);

}
```

40.23 HTMLHeadingElement.java

Secuencia 413: org/w3c/dom/html/HTMLHeadingElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  For the <code>H1</code> to <code>H6</code> elements. See the  H1 element
 * definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLHeadingElement extends HTMLElement {
    /**
     *  Horizontal text alignment. See the  align attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

}
```

40.24 HTMLHtmlElement.java

Secuencia 414: org/w3c/dom/html/HTMLHtmlElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Root of an HTML document. See the  HTML element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLHtmlElement extends HTMLElement {
    /**
     *  Version information about the document's DTD. See the  version
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getVersion();
    public void setVersion(String version);

}
```

40.25 HTMLIFrameElement.java

Secuencia 415: org/w3c/dom/html/HTMLIFrameElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.Document;

/**
 *  Inline subwindows. See the  IFRAME element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
```

```java
public interface HTMLIFrameElement extends HTMLElement {
    /**
     *  Aligns this object (vertically or horizontally)  with respect to its
     * surrounding text. See the  align attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Request frame borders. See the  frameborder attribute definition in
     * HTML 4.0.
     */
    public String getFrameBorder();
    public void setFrameBorder(String frameBorder);

    /**
     *  Frame height. See the  height attribute definition in HTML 4.0.
     */
    public String getHeight();
    public void setHeight(String height);

    /**
     *  URI designating a long description of this image or frame. See the
     * longdesc attribute definition in HTML 4.0.
     */
    public String getLongDesc();
    public void setLongDesc(String longDesc);

    /**
     *  Frame margin height, in pixels. See the  marginheight attribute
     * definition in HTML 4.0.
     */
    public String getMarginHeight();
    public void setMarginHeight(String marginHeight);

    /**
     *  Frame margin width, in pixels. See the  marginwidth attribute
     * definition in HTML 4.0.
     */
    public String getMarginWidth();
    public void setMarginWidth(String marginWidth);

    /**
     *  The frame name (object of the <code>target</code> attribute). See the
     * name attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Specify whether or not the frame should have scrollbars. See the
     * scrolling attribute definition in HTML 4.0.
```

```
     */
    public String getScrolling();
    public void setScrolling(String scrolling);

    /**
     *  A URI designating the initial frame contents. See the  src attribute
     * definition in HTML 4.0.
     */
    public String getSrc();
    public void setSrc(String src);

    /**
     *  Frame width. See the  width attribute definition in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

    /**
     *  The document this frame contains, if there is any and it is available,
     * or <code>null</code> otherwise.
     * @since DOM Level 2
     */
    public Document getContentDocument();

}
```

40.26 HTMLImageElement.java

Secuencia 416: org/w3c/dom/html/HTMLImageElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```java
/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Embedded image. See the  IMG element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLImageElement extends HTMLElement {
    /**
     *  URI designating the source of this image, for low-resolution output.
     */
    public String getLowSrc();
    public void setLowSrc(String lowSrc);

    /**
     *  The name of the element (for backwards compatibility).
     */
    public String getName();
    public void setName(String name);

    /**
     *  Aligns this object (vertically or horizontally)  with respect to its
     * surrounding text. See the  align attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Alternate text for user agents not rendering the normal content of
     * this element. See the  alt attribute definition in HTML 4.0.
     */
    public String getAlt();
    public void setAlt(String alt);

    /**
```

```
     *  Width of border around image. See the  border attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBorder();
    public void setBorder(String border);

    /**
     *  Override height. See the  height attribute definition in HTML 4.0.
     */
    public String getHeight();
    public void setHeight(String height);

    /**
     *  Horizontal space to the left and right of this image. See the  hspace
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getHspace();
    public void setHspace(String hspace);

    /**
     *  Use server-side image map. See the  ismap attribute definition in HTML
     * 4.0.
     */
    public boolean getIsMap();
    public void setIsMap(boolean isMap);

    /**
     *  URI designating a long description of this image or frame. See the
     * longdesc attribute definition in HTML 4.0.
     */
    public String getLongDesc();
    public void setLongDesc(String longDesc);

    /**
     *  URI designating the source of this image. See the  src attribute
     * definition in HTML 4.0.
     */
    public String getSrc();
    public void setSrc(String src);

    /**
     *  Use client-side image map. See the  usemap attribute definition in
     * HTML 4.0.
     */
    public String getUseMap();
    public void setUseMap(String useMap);

    /**
     *  Vertical space above and below this image. See the  vspace attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getVspace();
```

```
    public void setVspace(String vspace);

    /**
     *  Override width. See the  width attribute definition in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

}
```

40.27 HTMLInputElement.java

Secuencia 417: org/w3c/dom/html/HTMLInputElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */
```

```java

package org.w3c.dom.html;

/**
 *  Form control.  Note. Depending upon the environment in which the page is
 * being viewed, the value property may be read-only for the file upload
 * input type. For the "password" input type, the actual value returned may
 * be masked to prevent unauthorized use. See the  INPUT element definition
 * in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLInputElement extends HTMLElement {
    /**
     *  When the <code>type</code> attribute of the element has the value
     * "Text", "File" or "Password", this represents the HTML value attribute
     * of the element. The value of this attribute does not change if the
     * contents of the corresponding form control, in an interactive user
     * agent, changes. Changing this attribute, however, resets the contents
     * of the form control. See the  value attribute definition in HTML 4.0.
     */
    public String getDefaultValue();
    public void setDefaultValue(String defaultValue);

    /**
     *  When <code>type</code> has the value "Radio" or "Checkbox", this
     * represents the HTML checked attribute of the element. The value of
     * this attribute does not change if the state of the corresponding form
     * control, in an interactive user agent, changes. Changes to this
     * attribute, however, resets the state of the form control. See the
     * checked attribute definition in HTML 4.0.
     */
    public boolean getDefaultChecked();
    public void setDefaultChecked(boolean defaultChecked);

    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  A comma-separated list of content types that a server processing this
     * form will handle correctly. See the  accept attribute definition in
     * HTML 4.0.
     */
    public String getAccept();
    public void setAccept(String accept);

    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
```

```
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  Aligns this object (vertically or horizontally)  with respect to its
     * surrounding text. See the  align attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Alternate text for user agents not rendering the normal content of
     * this element. See the  alt attribute definition in HTML 4.0.
     */
    public String getAlt();
    public void setAlt(String alt);

    /**
     *  When the <code>type</code> attribute of the element has the value
     * "Radio" or "Checkbox", this represents the current state of the form
     * control, in an interactive user agent. Changes to this attribute
     * change the state of the form control, but do not change the value of
     * the HTML value attribute of the element.
     */
    public boolean getChecked();
    public void setChecked(boolean checked);

    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  Maximum number of characters for text fields, when <code>type</code>
     * has the value "Text" or "Password". See the  maxlength attribute
     * definition in HTML 4.0.
     */
    public int getMaxLength();
    public void setMaxLength(int maxLength);

    /**
     *  Form control or object name when submitted with a form. See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  This control is read-only. Relevant only when <code>type</code> has
     * the value "Text" or "Password". See the  readonly attribute definition
```

```
     * in HTML 4.0.
     */
    public boolean getReadOnly();
    public void setReadOnly(boolean readOnly);

    /**
     *  Size information. The precise meaning is specific to each type of
     * field.  See the  size attribute definition in HTML 4.0.
     */
    public String getSize();
    public void setSize(String size);

    /**
     *  When the <code>type</code> attribute has the value "Image", this
     * attribute specifies the location of the image to be used to decorate
     * the graphical submit button. See the  src attribute definition in HTML
     * 4.0.
     */
    public String getSrc();
    public void setSrc(String src);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  The type of control created. See the  type attribute definition in
     * HTML 4.0.
     */
    public String getType();

    /**
     *  Use client-side image map. See the  usemap attribute definition in
     * HTML 4.0.
     */
    public String getUseMap();
    public void setUseMap(String useMap);

    /**
     *  When the <code>type</code> attribute of the element has the value
     * "Text", "File" or "Password", this represents the current contents of
     * the corresponding form control, in an interactive user agent. Changing
     * this attribute changes the contents of the form control, but does not
     * change the value of the HTML value attribute of the element. When the
     * <code>type</code> attribute of the element has the value "Button",
     * "Hidden", "Submit", "Reset", "Image", "Checkbox" or "Radio", this
     * represents the HTML value attribute of the element. See the  value
     * attribute definition in HTML 4.0.
     */
    public String getValue();
```

```
    public void setValue(String value);

    /**
     *  Removes keyboard focus from this element.
     */
    public void blur();

    /**
     *  Gives keyboard focus to this element.
     */
    public void focus();

    /**
     *  Select the contents of the text area. For <code>INPUT</code> elements
     * whose <code>type</code> attribute has one of the following values:
     * "Text", "File", or "Password".
     */
    public void select();

    /**
     *  Simulate a mouse-click. For <code>INPUT</code> elements whose
     * <code>type</code> attribute has one of the following values: "Button",
     * "Checkbox", "Radio", "Reset", or "Submit".
     */
    public void click();

}
```

40.28 HTMLIsIndexElement.java

Secuencia 418: org/w3c/dom/html/HTMLIsIndexElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  This element is used for single-line text input. See the  ISINDEX element
 * definition in HTML 4.0. This element is deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLIsIndexElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  The prompt message. See the  prompt attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getPrompt();
    public void setPrompt(String prompt);

}
```

40.29 HTMLLIElement.java

Secuencia 419: org/w3c/dom/html/HTMLLIElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  List item. See the  LI element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLLIElement extends HTMLElement {
    /**
     *  List item bullet style. See the  type attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

    /**
     *  Reset sequence number when used in <code>OL</code> . See the  value
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
```

```
     */
    public int getValue();
    public void setValue(int value);

}
```

40.30 HTMLLabelElement.java

Secuencia 420: org/w3c/dom/html/HTMLLabelElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
```

```
 *  Form field label text. See the  LABEL element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLLabelElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  This attribute links this label with another form control by
     * <code>id</code> attribute. See the  for attribute definition in HTML
     * 4.0.
     */
    public String getHtmlFor();
    public void setHtmlFor(String htmlFor);

}
```

40.31 HTMLLegendElement.java

Secuencia 421: org/w3c/dom/html/HTMLLegendElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Provides a caption for a <code>FIELDSET</code> grouping.  See the  LEGEND
 * element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLLegendElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  Text alignment relative to <code>FIELDSET</code> . See the  align
     * attribute definition in HTML 4.0. This attribute is deprecated in HTML
     * 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

}
```

40.32 HTMLLinkElement.java

Secuencia 422: org/w3c/dom/html/HTMLLinkElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  The <code>LINK</code> element specifies a link to an external resource,
 * and defines this document's relationship to that resource (or vice versa).
 *  See the  LINK element definition in HTML 4.0  (see also the
 * <code>LinkStyle</code> interface in the  module).
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLLinkElement extends HTMLElement {
```

```
    /**
     *  Enables/disables the link. This is currently only used for style sheet
     * links, and may be used to activate or deactivate style sheets.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  The character encoding of the resource being linked to. See the
     * charset attribute definition in HTML 4.0.
     */
    public String getCharset();
    public void setCharset(String charset);

    /**
     *  The URI of the linked resource. See the  href attribute definition in
     * HTML 4.0.
     */
    public String getHref();
    public void setHref(String href);

    /**
     *  Language code of the linked resource. See the  hreflang attribute
     * definition in HTML 4.0.
     */
    public String getHreflang();
    public void setHreflang(String hreflang);

    /**
     *  Designed for use with one or more target media. See the  media
     * attribute definition in HTML 4.0.
     */
    public String getMedia();
    public void setMedia(String media);

    /**
     *  Forward link type. See the  rel attribute definition in HTML 4.0.
     */
    public String getRel();
    public void setRel(String rel);

    /**
     *  Reverse link type. See the  rev attribute definition in HTML 4.0.
     */
    public String getRev();
    public void setRev(String rev);

    /**
     *  Frame to render the resource in. See the  target attribute definition
     * in HTML 4.0.
     */
    public String getTarget();
    public void setTarget(String target);
```

```java

    /**
     *  Advisory content type. See the  type attribute definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

}
```

40.33 HTMLMapElement.java

Secuencia 423: org/w3c/dom/html/HTMLMapElement.java

```java
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

```

```
package org.w3c.dom.html;

/**
 *  Client-side image map. See the  MAP element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLMapElement extends HTMLElement {
    /**
     *  The list of areas defined for the image map.
     */
    public HTMLCollection getAreas();

    /**
     *  Names the map (for use with <code>usemap</code> ). See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

}
```

40.34 HTMLMenuElement.java

Secuencia 424: org/w3c/dom/html/HTMLMenuElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Menu list. See the  MENU element definition in HTML 4.0. This element is
 * deprecated in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLMenuElement extends HTMLElement {
    /**
     *  Reduce spacing between list items. See the  compact attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

40.35 HTMLMetaElement.java

Secuencia 425: org/w3c/dom/html/HTMLMetaElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  This contains generic meta-information about the document. See the  META
 * element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLMetaElement extends HTMLElement {
    /**
     *  Associated information. See the  content attribute definition in HTML
     * 4.0.
     */
    public String getContent();
    public void setContent(String content);

    /**
     *  HTTP response header name. See the  http-equiv attribute definition in
     * HTML 4.0.
     */
    public String getHttpEquiv();
    public void setHttpEquiv(String httpEquiv);

    /**
     *  Meta information name. See the  name attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Select form of content. See the  scheme attribute definition in HTML
```

```
     * 4.0.
     */
    public String getScheme();
    public void setScheme(String scheme);

}
```

40.36 HTMLModElement.java

Secuencia 426: org/w3c/dom/html/HTMLModElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

```

```
/**
 *  Notice of modification to part of a document. See the   INS  and  DEL
 * element definitions in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLModElement extends HTMLElement {
    /**
     *  A URI designating a document that describes the reason for the change.
     * See the  cite attribute definition in HTML 4.0.
     */
    public String getCite();
    public void setCite(String cite);

    /**
     *  The date and time of the change. See the  datetime attribute definition
     *  in HTML 4.0.
     */
    public String getDateTime();
    public void setDateTime(String dateTime);

}
```

40.37 HTMLOListElement.java

Secuencia 427: org/w3c/dom/html/HTMLOListElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
```

```
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Ordered list. See the  OL element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLOListElement extends HTMLElement {
    /**
     *  Reduce spacing between list items. See the  compact attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getCompact();
    public void setCompact(boolean compact);

    /**
     *  Starting sequence number. See the  start attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public int getStart();
    public void setStart(int start);

    /**
     *  Numbering style. See the  type attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

}
```

40.38 HTMLObjectElement.java

Secuencia 428: org/w3c/dom/html/HTMLObjectElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.Document;

/**
 *  Generic embedded object.  Note. In principle, all properties on the object
 * element are read-write but in some environments some properties may be
 * read-only once the underlying object is instantiated. See the  OBJECT
 * element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLObjectElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
```

```
    public HTMLFormElement getForm();

    /**
     *  Applet class file. See the <code>code</code> attribute for
     * HTMLAppletElement.
     */
    public String getCode();
    public void setCode(String code);

    /**
     *  Aligns this object (vertically or horizontally)  with respect to its
     * surrounding text. See the  align attribute definition in HTML 4.0.
     * This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Space-separated list of archives. See the  archive attribute definition
     *  in HTML 4.0.
     */
    public String getArchive();
    public void setArchive(String archive);

    /**
     *  Width of border around the object. See the  border attribute definition
     *  in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBorder();
    public void setBorder(String border);

    /**
     *  Base URI for <code>classid</code> , <code>data</code> , and
     * <code>archive</code> attributes. See the  codebase attribute definition
     *  in HTML 4.0.
     */
    public String getCodeBase();
    public void setCodeBase(String codeBase);

    /**
     *  Content type for data downloaded via <code>classid</code> attribute.
     * See the  codetype attribute definition in HTML 4.0.
     */
    public String getCodeType();
    public void setCodeType(String codeType);

    /**
     *  A URI specifying the location of the object's data.  See the  data
     * attribute definition in HTML 4.0.
     */
    public String getData();
    public void setData(String data);

```

```
    /**
     *  Declare (for future reference), but do not instantiate, this object.
     * See the  declare attribute definition in HTML 4.0.
     */
    public boolean getDeclare();
    public void setDeclare(boolean declare);

    /**
     *  Override height. See the  height attribute definition in HTML 4.0.
     */
    public String getHeight();
    public void setHeight(String height);

    /**
     *  Horizontal space to the left and right of this image, applet, or
     * object. See the  hspace attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getHspace();
    public void setHspace(String hspace);

    /**
     *  Form control or object name when submitted with a form. See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Message to render while loading the object. See the  standby attribute
     * definition in HTML 4.0.
     */
    public String getStandby();
    public void setStandby(String standby);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  Content type for data downloaded via <code>data</code> attribute. See
     * the  type attribute definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

    /**
     *  Use client-side image map. See the  usemap attribute definition in
     * HTML 4.0.
     */
```

```
    public String getUseMap();
    public void setUseMap(String useMap);

    /**
     *  Vertical space above and below this image, applet, or object. See the
     * vspace attribute definition in HTML 4.0. This attribute is deprecated
     * in HTML 4.0.
     */
    public String getVspace();
    public void setVspace(String vspace);

    /**
     *  Override width. See the  width attribute definition in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

    /**
     *  The document this object contains, if there is any and it is
     * available, or <code>null</code> otherwise.
     * @since DOM Level 2
     */
    public Document getContentDocument();

}
```

40.39 HTMLOptGroupElement.java

Secuencia 429: org/w3c/dom/html/HTMLOptGroupElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```
/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Group options together in logical subdivisions. See the  OPTGROUP element
 * definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLOptGroupElement extends HTMLElement {
    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  Assigns a label to this option group. See the  label attribute
     * definition in HTML 4.0.
     */
    public String getLabel();
    public void setLabel(String label);

}
```

40.40 HTMLOptionElement.java

Secuencia 430: org/w3c/dom/html/HTMLOptionElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  A selectable choice. See the  OPTION element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLOptionElement extends HTMLElement {
    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  Represents the value of the HTML selected attribute. The value of this
     * attribute does not change if the state of the corresponding form
     * control, in an interactive user agent, changes. Changing
     * <code>defaultSelected</code> , however, resets the state of the form
     * control. See the  selected attribute definition in HTML 4.0.
```

```
     */
    public boolean getDefaultSelected();
    public void setDefaultSelected(boolean defaultSelected);

    /**
     *  The text contained within the option element.
     */
    public String getText();

    /**
     *  The index of this <code>OPTION</code> in its parent <code>SELECT</code>
     *  , starting from 0.
     */
    public int getIndex();

    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  Option label for use in hierarchical menus. See the  label attribute
     * definition in HTML 4.0.
     */
    public String getLabel();
    public void setLabel(String label);

    /**
     *  Represents the current state of the corresponding form control, in an
     * interactive user agent. Changing this attribute changes the state of
     * the form control, but does not change the value of the HTML selected
     * attribute of the element.
     */
    public boolean getSelected();
    public void setSelected(boolean selected);

    /**
     *  The current form control value. See the  value attribute definition in
     * HTML 4.0.
     */
    public String getValue();
    public void setValue(String value);

}
```

40.41 HTMLParagraphElement.java

Secuencia 431: org/w3c/dom/html/HTMLParagraphElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Paragraphs. See the  P element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLParagraphElement extends HTMLElement {
    /**
     *  Horizontal text alignment. See the  align attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

```

```
}
```

40.42 HTMLParamElement.java

Secuencia 432: org/w3c/dom/html/HTMLParamElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Parameters fed to the <code>OBJECT</code> element. See the  PARAM element
 * definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
```

```
 */
public interface HTMLParamElement extends HTMLElement {
    /**
     *  The name of a run-time parameter. See the  name attribute definition
     * in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  Content type for the <code>value</code> attribute when
     * <code>valuetype</code> has the value "ref". See the  type attribute
     * definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

    /**
     *  The value of a run-time parameter. See the  value attribute definition
     * in HTML 4.0.
     */
    public String getValue();
    public void setValue(String value);

    /**
     *  Information about the meaning of the <code>value</code> attribute
     * value. See the  valuetype attribute definition in HTML 4.0.
     */
    public String getValueType();
    public void setValueType(String valueType);

}
```

40.43 HTMLPreElement.java

Secuencia 433: org/w3c/dom/html/HTMLPreElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Preformatted text. See the  PRE element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLPreElement extends HTMLElement {
    /**
     *  Fixed width for content. See the  width attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public int getWidth();
    public void setWidth(int width);

}
```

40.44 HTMLQuoteElement.java

Secuencia 434: org/w3c/dom/html/HTMLQuoteElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  For the <code>Q</code> and <code>BLOCKQUOTE</code> elements. See the  Q
 * element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLQuoteElement extends HTMLElement {
    /**
     *  A URI designating a source document or message. See the  cite
     * attribute definition in HTML 4.0.
     */
    public String getCite();
    public void setCite(String cite);

}
```

40.45 HTMLScriptElement.java

Secuencia 435: org/w3c/dom/html/HTMLScriptElement.java

```java
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Script statements. See the  SCRIPT element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLScriptElement extends HTMLElement {
    /**
     *  The script content of the element.
     */
    public String getText();
```

```
    public void setText(String text);

    /**
     *  Reserved for future use.
     */
    public String getHtmlFor();
    public void setHtmlFor(String htmlFor);

    /**
     *  Reserved for future use.
     */
    public String getEvent();
    public void setEvent(String event);

    /**
     *  The character encoding of the linked resource. See the  charset
     * attribute definition in HTML 4.0.
     */
    public String getCharset();
    public void setCharset(String charset);

    /**
     *  Indicates that the user agent can defer processing of the script.  See
     * the  defer attribute definition in HTML 4.0.
     */
    public boolean getDefer();
    public void setDefer(boolean defer);

    /**
     *  URI designating an external script. See the  src attribute definition
     * in HTML 4.0.
     */
    public String getSrc();
    public void setSrc(String src);

    /**
     *  The content type of the script language. See the  type attribute
     * definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

}
```

40.46 HTMLSelectElement.java

Secuencia 436: org/w3c/dom/html/HTMLSelectElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.DOMException;

/**
 *  The select element allows the selection of an option. The contained
 * options can be directly accessed through the select element as a
 * collection. See the  SELECT element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLSelectElement extends HTMLElement {
    /**
     *  The type of this form control. This is the string "select-multiple"
     * when the multiple attribute is <code>true</code> and the string
     * "select-one" when <code>false</code> .
     */
    public String getType();
```

```java

    /**
     *  The ordinal index of the selected option, starting from 0. The value
     * -1 is returned if no element is selected. If multiple options are
     * selected, the index of the first selected option is returned.
     */
    public int getSelectedIndex();
    public void setSelectedIndex(int selectedIndex);

    /**
     *  The current form control value.
     */
    public String getValue();
    public void setValue(String value);

    /**
     *  The number of options in this <code>SELECT</code> .
     */
    public int getLength();

    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  The collection of <code>OPTION</code> elements contained by this
     * element.
     */
    public HTMLCollection getOptions();

    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  If true, multiple <code>OPTION</code> elements may  be selected in
     * this <code>SELECT</code> . See the  multiple attribute definition in
     * HTML 4.0.
     */
    public boolean getMultiple();
    public void setMultiple(boolean multiple);

    /**
     *  Form control or object name when submitted with a form. See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);
```

```

    /**
     *  Number of visible rows. See the  size attribute definition in HTML 4.0.
     */
    public int getSize();
    public void setSize(int size);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  Add a new element to the collection of <code>OPTION</code> elements
     * for this <code>SELECT</code> . This method is the equivalent of the
     * <code>appendChild</code> method of the <code>Node</code> interface if
     * the <code>before</code> parameter is <code>null</code> . It is
     * equivalent to the <code>insertBefore</code> method on the parent of
     * <code>before</code> in all other cases.
     * @param element  The element to add.
     * @param before  The element to insert before, or <code>null</code> for
     *   the tail of the list.
     * @exception DOMException
     *    NOT_FOUND_ERR: Raised if <code>before</code> is not a descendant of
     *   the <code>SELECT</code> element.
     */
    public void add(HTMLElement element,
                    HTMLElement before)
                    throws DOMException;

    /**
     *  Remove an element from the collection of <code>OPTION</code> elements
     * for this <code>SELECT</code> . Does nothing if no element has the given
     *  index.
     * @param index  The index of the item to remove, starting from 0.
     */
    public void remove(int index);

    /**
     *  Removes keyboard focus from this element.
     */
    public void blur();

    /**
     *  Gives keyboard focus to this element.
     */
    public void focus();

}
```

40.47 HTMLStyleElement.java

Secuencia 437: org/w3c/dom/html/HTMLStyleElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Style information. See the  STYLE element definition in HTML 4.0, the
 * module and the <code>LinkStyle</code> interface in the  module.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLStyleElement extends HTMLElement {
```

```
    /**
     *  Enables/disables the style sheet.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    /**
     *  Designed for use with one or more target media. See the  media
     * attribute definition in HTML 4.0.
     */
    public String getMedia();
    public void setMedia(String media);

    /**
     *  The content type pf the style sheet language. See the  type attribute
     * definition in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

}
```

40.48 HTMLTableCaptionElement.java

Secuencia 438: org/w3c/dom/html/HTMLTableCaptionElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
```

```
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Table caption See the  CAPTION element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableCaptionElement extends HTMLElement {
    /**
     *  Caption alignment with respect to the table. See the  align attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

}
```

40.49 HTMLTableCellElement.java

Secuencia 439: org/w3c/dom/html/HTMLTableCellElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  The object used to represent the <code>TH</code> and <code>TD</code>
 * elements. See the  TD element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableCellElement extends HTMLElement {
    /**
     *  The index of this cell in the row, starting from 0. This index is in
     * document tree order and not display order.
     */
    public int getCellIndex();

    /**
     *  Abbreviation for header cells. See the  abbr attribute definition in
     * HTML 4.0.
     */
    public String getAbbr();
    public void setAbbr(String abbr);

    /**
     *  Horizontal alignment of data in cell. See the  align attribute
     * definition in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Names group of related headers. See the  axis attribute definition in
     * HTML 4.0.
```

```
     */
    public String getAxis();
    public void setAxis(String axis);

    /**
     *  Cell background color. See the  bgcolor attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBgColor();
    public void setBgColor(String bgColor);

    /**
     *  Alignment character for cells in a column. See the  char attribute
     * definition in HTML 4.0.
     */
    public String getCh();
    public void setCh(String ch);

    /**
     *  Offset of alignment character. See the  charoff attribute definition
     * in HTML 4.0.
     */
    public String getChOff();
    public void setChOff(String chOff);

    /**
     *  Number of columns spanned by cell. See the  colspan attribute
     * definition in HTML 4.0.
     */
    public int getColSpan();
    public void setColSpan(int colSpan);

    /**
     *  List of <code>id</code> attribute values for header cells. See the
     * headers attribute definition in HTML 4.0.
     */
    public String getHeaders();
    public void setHeaders(String headers);

    /**
     *  Cell height. See the  height attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getHeight();
    public void setHeight(String height);

    /**
     *  Suppress word wrapping. See the  nowrap attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getNoWrap();
    public void setNoWrap(boolean noWrap);

```

```
    /**
     *  Number of rows spanned by cell. See the  rowspan attribute definition
     * in HTML 4.0.
     */
    public int getRowSpan();
    public void setRowSpan(int rowSpan);

    /**
     *  Scope covered by header cells. See the  scope attribute definition in
     * HTML 4.0.
     */
    public String getScope();
    public void setScope(String scope);

    /**
     *  Vertical alignment of data in cell. See the  valign attribute
     * definition in HTML 4.0.
     */
    public String getVAlign();
    public void setVAlign(String vAlign);

    /**
     *  Cell width. See the  width attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

}
```

40.50 HTMLTableColElement.java

Secuencia 440: org/w3c/dom/html/HTMLTableColElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Regroups the <code>COL</code> and <code>COLGROUP</code> elements. See the
 * COL element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableColElement extends HTMLElement {
    /**
     *  Horizontal alignment of cell data in column. See the  align attribute
     * definition in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Alignment character for cells in a column. See the  char attribute
     * definition in HTML 4.0.
     */
    public String getCh();
    public void setCh(String ch);

    /**
     *  Offset of alignment character. See the  charoff attribute definition
     * in HTML 4.0.
     */
    public String getChOff();
    public void setChOff(String chOff);

    /**
     *  Indicates the number of columns in a group or affected by a grouping.
```

```
     * See the  span attribute definition in HTML 4.0.
     */
    public int getSpan();
    public void setSpan(int span);

    /**
     *  Vertical alignment of cell data in column. See the  valign attribute
     * definition in HTML 4.0.
     */
    public String getVAlign();
    public void setVAlign(String vAlign);

    /**
     *  Default column width. See the  width attribute definition in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

}
```

40.51 HTMLTableElement.java

Secuencia 441: org/w3c/dom/html/HTMLTableElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
```

```
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.DOMException;

/**
 *  The create* and delete* methods on the table allow authors to construct
 * and modify tables. HTML 4.0 specifies that only one of each of the
 * <code>CAPTION</code> , <code>THEAD</code> , and <code>TFOOT</code>
 * elements may exist in a table. Therefore, if one exists, and the
 * createTHead() or createTFoot() method is called, the method returns the
 * existing THead or TFoot element. See the  TABLE element definition in HTML
 * 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableElement extends HTMLElement {
    /**
     *  Returns the table's <code>CAPTION</code> , or void if none exists.
     */
    public HTMLTableCaptionElement getCaption();
    public void setCaption(HTMLTableCaptionElement caption);

    /**
     *  Returns the table's <code>THEAD</code> , or <code>null</code> if none
     * exists.
     */
    public HTMLTableSectionElement getTHead();
    public void setTHead(HTMLTableSectionElement tHead);

    /**
     *  Returns the table's <code>TFOOT</code> , or <code>null</code> if none
     * exists.
     */
    public HTMLTableSectionElement getTFoot();
    public void setTFoot(HTMLTableSectionElement tFoot);

    /**
     *  Returns a collection of all the rows in the table, including all in
     * <code>THEAD</code> , <code>TFOOT</code> , all <code>TBODY</code>
     * elements.
     */
    public HTMLCollection getRows();
```

```

    /**
     *  Returns a collection of the defined table bodies.
     */
    public HTMLCollection getTBodies();

    /**
     *  Specifies the table's position with respect to the rest of the
     * document. See the  align attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Cell background color. See the  bgcolor attribute definition in HTML
     * 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBgColor();
    public void setBgColor(String bgColor);

    /**
     *  The width of the border around the table. See the  border attribute
     * definition in HTML 4.0.
     */
    public String getBorder();
    public void setBorder(String border);

    /**
     *  Specifies the horizontal and vertical space between cell content and
     * cell borders. See the  cellpadding attribute definition in HTML 4.0.
     */
    public String getCellPadding();
    public void setCellPadding(String cellPadding);

    /**
     *  Specifies the horizontal and vertical separation between cells. See
     * the  cellspacing attribute definition in HTML 4.0.
     */
    public String getCellSpacing();
    public void setCellSpacing(String cellSpacing);

    /**
     *  Specifies which external table borders to render. See the  frame
     * attribute definition in HTML 4.0.
     */
    public String getFrame();
    public void setFrame(String frame);

    /**
     *  Specifies which internal table borders to render. See the  rules
     * attribute definition in HTML 4.0.
     */
```

```
    public String getRules();
    public void setRules(String rules);

    /**
     *  Description about the purpose or structure of a table. See the
     * summary attribute definition in HTML 4.0.
     */
    public String getSummary();
    public void setSummary(String summary);

    /**
     *  Specifies the desired table width. See the  width attribute definition
     * in HTML 4.0.
     */
    public String getWidth();
    public void setWidth(String width);

    /**
     *  Create a table header row or return an existing one.
     * @return  A new table header element (<code>THEAD</code> ).
     */
    public HTMLElement createTHead();

    /**
     *  Delete the header from the table, if one exists.
     */
    public void deleteTHead();

    /**
     *  Create a table footer row or return an existing one.
     * @return  A footer element (<code>TFOOT</code> ).
     */
    public HTMLElement createTFoot();

    /**
     *  Delete the footer from the table, if one exists.
     */
    public void deleteTFoot();

    /**
     *  Create a new table caption object or return an existing one.
     * @return  A <code>CAPTION</code> element.
     */
    public HTMLElement createCaption();

    /**
     *  Delete the table caption, if one exists.
     */
    public void deleteCaption();

    /**
     *  Insert a new empty row in the table. The new row is inserted
     * immediately before and in the same section as the current
```

```
     * <code>index</code> th row in the table. If <code>index</code> is equal
     * to the number of rows, the new row is appended. In addition, when the
     * table is empty the row is inserted into a <code>TBODY</code> which is
     * created and inserted into the table. Note. A table row cannot be empty
     * according to HTML 4.0 Recommendation.
     * @param index  The row number where to insert a new row. This index
     *   starts from 0 and is relative to all the rows contained inside the
     *   table, regardless of section parentage.
     * @return  The newly created row.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified index is greater than the
     *   number of rows or if the index is negative.
     */
    public HTMLElement insertRow(int index)
                                 throws DOMException;

    /**
     *  Delete a table row.
     * @param index  The index of the row to be deleted. This index starts
     *   from 0 and is relative to all the rows contained inside the table,
     *   regardless of section parentage.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified index is greater than or
     *   equal to the number of rows or if the index is negative.
     */
    public void deleteRow(int index)
                          throws DOMException;

}
```

40.52 HTMLTableRowElement.java

Secuencia 442: org/w3c/dom/html/HTMLTableRowElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

import org.w3c.dom.DOMException;

/**
 *  A row in a table. See the  TR element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableRowElement extends HTMLElement {
    /**
     *  The index of this row, relative to the entire table, starting from 0.
     * This is in document tree order and not display order. The
     * <code>rowIndex</code> does not take into account sections (
     * <code>THEAD</code> , <code>TFOOT</code> , or <code>TBODY</code> )
     * within the table.
     */
    public int getRowIndex();

    /**
     *  The index of this row, relative to the current section (
     * <code>THEAD</code> , <code>TFOOT</code> , or <code>TBODY</code> ),
     * starting from 0.
     */
    public int getSectionRowIndex();

    /**
     *  The collection of cells in this row.
     */
    public HTMLCollection getCells();

    /**
```

```
     *  Horizontal alignment of data within cells of this row. See the  align
     * attribute definition in HTML 4.0.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Background color for rows. See the  bgcolor attribute definition in
     * HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public String getBgColor();
    public void setBgColor(String bgColor);

    /**
     *  Alignment character for cells in a column. See the  char attribute
     * definition in HTML 4.0.
     */
    public String getCh();
    public void setCh(String ch);

    /**
     *  Offset of alignment character. See the  charoff attribute definition
     * in HTML 4.0.
     */
    public String getChOff();
    public void setChOff(String chOff);

    /**
     *  Vertical alignment of data within cells of this row. See the  valign
     * attribute definition in HTML 4.0.
     */
    public String getVAlign();
    public void setVAlign(String vAlign);

    /**
     *  Insert an empty <code>TD</code> cell into this row. If
     * <code>index</code> is equal to the number of cells, the new cell is
     * appended
     * @param index  The place to insert the cell, starting from 0.
     * @return  The newly created cell.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified <code>index</code> is
     *   greater than the number of cells or if the index is negative.
     */
    public HTMLElement insertCell(int index)
                                  throws DOMException;

    /**
     *  Delete a cell from the current row.
     * @param index  The index of the cell to delete, starting from 0.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified <code>index</code> is
     *   greater than or equal to the number of cells or if the index is
```

```
     *   negative.
     */
    public void deleteCell(int index)
                           throws DOMException;

}
```

40.53 HTMLTableSectionElement.java

Secuencia 443: org/w3c/dom/html/HTMLTableSectionElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

```

```java
import org.w3c.dom.DOMException;

/**
 *  The <code>THEAD</code> , <code>TFOOT</code> , and <code>TBODY</code>
 * elements.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTableSectionElement extends HTMLElement {
    /**
     *  Horizontal alignment of data in cells. See the <code>align</code>
     * attribute for HTMLTheadElement for details.
     */
    public String getAlign();
    public void setAlign(String align);

    /**
     *  Alignment character for cells in a column. See the  char attribute
     * definition in HTML 4.0.
     */
    public String getCh();
    public void setCh(String ch);

    /**
     *  Offset of alignment character. See the  charoff attribute definition
     * in HTML 4.0.
     */
    public String getChOff();
    public void setChOff(String chOff);

    /**
     *  Vertical alignment of data in cells. See the <code>valign</code>
     * attribute for HTMLTheadElement for details.
     */
    public String getVAlign();
    public void setVAlign(String vAlign);

    /**
     *  The collection of rows in this table section.
     */
    public HTMLCollection getRows();

    /**
     *  Insert a row into this section. The new row is inserted immediately
     * before the current <code>index</code> th row in this section. If
     * <code>index</code> is equal to the number of rows in this section, the
     * new row is appended.
     * @param index  The row number where to insert a new row. This index
     *   starts from 0 and is relative only to the rows contained inside this
     *   section, not all the rows in the table.
     * @return  The newly created row.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified index is greater than the
```

```
     *   number of rows of if the index is neagative.
     */
    public HTMLElement insertRow(int index)
                                 throws DOMException;

    /**
     *  Delete a row from this section.
     * @param index  The index of the row to be deleted. This index starts
     *   from 0 and is relative only to the rows contained inside this
     *   section, not all the rows in the table.
     * @exception DOMException
     *    INDEX_SIZE_ERR: Raised if the specified index is greater than or
     *   equal to the number of rows or if the index is negative.
     */
    public void deleteRow(int index)
                          throws DOMException;

}
```

40.54 HTMLTextAreaElement.java

Secuencia 444: org/w3c/dom/html/HTMLTextAreaElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
```

```
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Multi-line text field. See the  TEXTAREA element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTextAreaElement extends HTMLElement {
    /**
     *  Represents the contents of the element. The value of this attribute
     * does not change if the contents of the corresponding form control, in
     * an interactive user agent, changes. Changing this attribute, however,
     * resets the contents of the form control.
     */
    public String getDefaultValue();
    public void setDefaultValue(String defaultValue);

    /**
     *  Returns the <code>FORM</code> element containing this control. Returns
     * <code>null</code> if this control is not within the context of a form.
     */
    public HTMLFormElement getForm();

    /**
     *  A single character access key to give access to the form control. See
     * the  accesskey attribute definition in HTML 4.0.
     */
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    /**
     *  Width of control (in characters). See the  cols attribute definition
     * in HTML 4.0.
     */
    public int getCols();
    public void setCols(int cols);

    /**
     *  The control is unavailable in this context. See the  disabled
     * attribute definition in HTML 4.0.
     */
    public boolean getDisabled();
    public void setDisabled(boolean disabled);
```

```java

    /**
     *  Form control or object name when submitted with a form. See the  name
     * attribute definition in HTML 4.0.
     */
    public String getName();
    public void setName(String name);

    /**
     *  This control is read-only. See the  readonly attribute definition in
     * HTML 4.0.
     */
    public boolean getReadOnly();
    public void setReadOnly(boolean readOnly);

    /**
     *  Number of text rows. See the  rows attribute definition in HTML 4.0.
     */
    public int getRows();
    public void setRows(int rows);

    /**
     *  Index that represents the element's position in the tabbing order. See
     * the  tabindex attribute definition in HTML 4.0.
     */
    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    /**
     *  The type of this form control. This the string "textarea".
     */
    public String getType();

    /**
     *  Represents the current contents of the corresponding form control, in
     * an interactive user agent. Changing this attribute changes the
     * contents of the form control, but does not change the contents of the
     * element. If the entirety of the data can not fit into a single
     * <code>DOMString</code> , the implementation may truncate the data.
     */
    public String getValue();
    public void setValue(String value);

    /**
     *  Removes keyboard focus from this element.
     */
    public void blur();

    /**
     *  Gives keyboard focus to this element.
     */
    public void focus();

```

```
    /**
     *  Select the contents of the <code>TEXTAREA</code> .
     */
    public void select();

}
```

40.55 HTMLTitleElement.java

Secuencia 445: org/w3c/dom/html/HTMLTitleElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

```

```
/**
 *  The document title. See the  TITLE element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLTitleElement extends HTMLElement {
    /**
     *  The specified title as a string.
     */
    public String getText();
    public void setText(String text);

}
```

40.56 HTMLUListElement.java

Secuencia 446: org/w3c/dom/html/HTMLUListElement.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
```

```
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE. See W3C License http://www.w3.org/Consortium/Legal/ for more
 * details.
 */

package org.w3c.dom.html;

/**
 *  Unordered list. See the  UL element definition in HTML 4.0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510'>Document Object
     Model (DOM) Level 2 Specification</a>.
 */
public interface HTMLUListElement extends HTMLElement {
    /**
     *  Reduce spacing between list items. See the  compact attribute
     * definition in HTML 4.0. This attribute is deprecated in HTML 4.0.
     */
    public boolean getCompact();
    public void setCompact(boolean compact);

    /**
     *  Bullet style. See the  type attribute definition in HTML 4.0. This
     * attribute is deprecated in HTML 4.0.
     */
    public String getType();
    public void setType(String type);

}
```

41 Dom Ls (org.w3c.dom.ls package)

41.1 DOMImplementationLS.java

Secuencia 447: org/w3c/dom/ls/DOMImplementationLS.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

import org.w3c.dom.DOMException;

/**
 *  <code>DOMImplementationLS</code> contains the factory methods for creating
 * Load and Save objects.
 * <p> The expectation is that an instance of the
 * <code>DOMImplementationLS</code> interface can be obtained by using
 * binding-specific casting methods on an instance of the
 * <code>DOMImplementation</code> interface or, if the <code>Document</code>
 * supports the feature <code>"Core"</code> version <code>"3.0"</code>
 * defined in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
 * , by using the method <code>DOMImplementation.getFeature</code> with
 * parameter values <code>"LS"</code> (or <code>"LS-Async"</code>) and
 * <code>"3.0"</code> (respectively).
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface DOMImplementationLS {
    // DOMImplementationLSMode
    /**
     * Create a synchronous <code>LSParser</code>.
     */
    public static final short MODE_SYNCHRONOUS          = 1;
    /**
     * Create an asynchronous <code>LSParser</code>.
     */
    public static final short MODE_ASYNCHRONOUS         = 2;
```

```

    /**
     * Create a new <code>LSParser</code>. The newly constructed parser may
     * then be configured by means of its <code>DOMConfiguration</code>
     * object, and used to parse documents by means of its <code>parse</code>
     *  method.
     * @param mode  The <code>mode</code> argument is either
     *   <code>MODE_SYNCHRONOUS</code> or <code>MODE_ASYNCHRONOUS</code>, if
     *   <code>mode</code> is <code>MODE_SYNCHRONOUS</code> then the
     *   <code>LSParser</code> that is created will operate in synchronous
     *   mode, if it's <code>MODE_ASYNCHRONOUS</code> then the
     *   <code>LSParser</code> that is created will operate in asynchronous
     *   mode.
     * @param schemaType  An absolute URI representing the type of the schema
     *   language used during the load of a <code>Document</code> using the
     *   newly created <code>LSParser</code>. Note that no lexical checking
     *   is done on the absolute URI. In order to create a
     *   <code>LSParser</code> for any kind of schema types (i.e. the
     *   LSParser will be free to use any schema found), use the value
     *   <code>null</code>.
     * <p ><b>Note:</b>    For W3C XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema
         -1-20010502/'>XML Schema Part 1</a>]
     *   , applications must use the value
     *   <code>"http://www.w3.org/2001/XMLSchema"</code>. For XML DTD [<a href='http://www.w3.org/
         TR/2004/REC-xml-20040204'>XML 1.0</a>],
     *   applications must use the value
     *   <code>"http://www.w3.org/TR/REC-xml"</code>. Other Schema languages
     *   are outside the scope of the W3C and therefore should recommend an
     *   absolute URI in order to use this method.
     * @return  The newly created <code>LSParser</code> object. This
     *   <code>LSParser</code> is either synchronous or asynchronous
     *   depending on the value of the <code>mode</code> argument.
     * <p ><b>Note:</b>    By default, the newly created <code>LSParser</code>
     *   does not contain a <code>DOMErrorHandler</code>, i.e. the value of
     *   the "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" configuration parameter is <code>null</code>. However, implementations
     *   may provide a default error handler at creation time. In that case,
     *   the initial value of the <code>"error-handler"</code> configuration
     *   parameter on the new <code>LSParser</code> object contains a
     *   reference to the default error handler.
     * @exception DOMException
     *    NOT_SUPPORTED_ERR: Raised if the requested mode or schema type is
     *   not supported.
     */
    public LSParser createLSParser(short mode,
                                   String schemaType)
                                   throws DOMException;

    /**
     *  Create a new <code>LSSerializer</code> object.
     * @return The newly created <code>LSSerializer</code> object.
     * <p ><b>Note:</b>    By default, the newly created
     *   <code>LSSerializer</code> has no <code>DOMErrorHandler</code>, i.e.
```

```
     *   the value of the <code>"error-handler"</code> configuration
     *   parameter is <code>null</code>. However, implementations may
     *   provide a default error handler at creation time. In that case, the
     *   initial value of the <code>"error-handler"</code> configuration
     *   parameter on the new <code>LSSerializer</code> object contains a
     *   reference to the default error handler.
     */
    public LSSerializer createLSSerializer();

    /**
     *  Create a new empty input source object where
     * <code>LSInput.characterStream</code>, <code>LSInput.byteStream</code>
     * , <code>LSInput.stringData</code> <code>LSInput.systemId</code>,
     * <code>LSInput.publicId</code>, <code>LSInput.baseURI</code>, and
     * <code>LSInput.encoding</code> are null, and
     * <code>LSInput.certifiedText</code> is false.
     * @return  The newly created input object.
     */
    public LSInput createLSInput();

    /**
     *  Create a new empty output destination object where
     * <code>LSOutput.characterStream</code>,
     * <code>LSOutput.byteStream</code>, <code>LSOutput.systemId</code>,
     * <code>LSOutput.encoding</code> are null.
     * @return  The newly created output object.
     */
    public LSOutput createLSOutput();

}
```

41.2 LSException.java

Secuencia 448: org/w3c/dom/ls/LSException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

/**
 *  Parser or write operations may throw an <code>LSException</code> if the
 * processing is stopped. The processing can be stopped due to a
 * <code>DOMError</code> with a severity of
 * <code>DOMError.SEVERITY_FATAL_ERROR</code> or a non recovered
 * <code>DOMError.SEVERITY_ERROR</code>, or if
 * <code>DOMErrorHandler.handleError()</code> returned <code>false</code>.
 * <p ><b>Note:</b>  As suggested in the definition of the constants in the
 * <code>DOMError</code> interface, a DOM implementation may choose to
 * continue after a fatal error, but the resulting DOM tree is then
 * implementation dependent.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public class LSException extends RuntimeException {
    public LSException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
    // LSExceptionCode
    /**
     *  If an attempt was made to load a document, or an XML Fragment, using
     * <code>LSParser</code> and the processing has been stopped.
     */
    public static final short PARSE_ERR                 = 81;
    /**
     *  If an attempt was made to serialize a <code>Node</code> using
```

```
     * <code>LSSerializer</code> and the processing has been stopped.
     */
    public static final short SERIALIZE_ERR            = 82;

}
```

41.3 LSInput.java

Secuencia 449: org/w3c/dom/ls/LSInput.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

/**
```

```
 *  This interface represents an input source for data.
 * <p> This interface allows an application to encapsulate information about
 * an input source in a single object, which may include a public
 * identifier, a system identifier, a byte stream (possibly with a specified
 * encoding), a base URI, and/or a character stream.
 * <p> The exact definitions of a byte stream and a character stream are
 * binding dependent.
 * <p> The application is expected to provide objects that implement this
 * interface whenever such objects are needed. The application can either
 * provide its own objects that implement this interface, or it can use the
 * generic factory method <code>DOMImplementationLS.createLSInput()</code>
 * to create objects that implement this interface.
 * <p> The <code>LSParser</code> will use the <code>LSInput</code> object to
 * determine how to read data. The <code>LSParser</code> will look at the
 * different inputs specified in the <code>LSInput</code> in the following
 * order to know which one to read from, the first one that is not null and
 * not an empty string will be used:
 * <ol>
 * <li> <code>LSInput.characterStream</code>
 * </li>
 * <li>
 * <code>LSInput.byteStream</code>
 * </li>
 * <li> <code>LSInput.stringData</code>
 * </li>
 * <li>
 * <code>LSInput.systemId</code>
 * </li>
 * <li> <code>LSInput.publicId</code>
 * </li>
 * </ol>
 * <p> If all inputs are null, the <code>LSParser</code> will report a
 * <code>DOMError</code> with its <code>DOMError.type</code> set to
 * <code>"no-input-specified"</code> and its <code>DOMError.severity</code>
 * set to <code>DOMError.SEVERITY_FATAL_ERROR</code>.
 * <p> <code>LSInput</code> objects belong to the application. The DOM
 * implementation will never modify them (though it may make copies and
 * modify the copies, if necessary).
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSInput {
    /**
     *  An attribute of a language and binding dependent type that represents
     * a stream of 16-bit units. The application must encode the stream
     * using UTF-16 (defined in [Unicode] and in [ISO/IEC 10646]). It is not a requirement to have
         an XML declaration when
     * using character streams. If an XML declaration is present, the value
     * of the encoding attribute will be ignored.
     */
    public java.io.Reader getCharacterStream();
    /**
```

```
     *  An attribute of a language and binding dependent type that represents
     * a stream of 16-bit units. The application must encode the stream
     * using UTF-16 (defined in [Unicode] and in [ISO/IEC 10646]). It is not a requirement to have an XML declaration when
     * using character streams. If an XML declaration is present, the value
     * of the encoding attribute will be ignored.
     */
    public void setCharacterStream(java.io.Reader characterStream);

    /**
     *  An attribute of a language and binding dependent type that represents
     * a stream of bytes.
     * <br> If the application knows the character encoding of the byte
     * stream, it should set the encoding attribute. Setting the encoding in
     * this way will override any encoding specified in an XML declaration
     * in the data.
     */
    public java.io.InputStream getByteStream();
    /**
     *  An attribute of a language and binding dependent type that represents
     * a stream of bytes.
     * <br> If the application knows the character encoding of the byte
     * stream, it should set the encoding attribute. Setting the encoding in
     * this way will override any encoding specified in an XML declaration
     * in the data.
     */
    public void setByteStream(java.io.InputStream byteStream);

    /**
     *  String data to parse. If provided, this will always be treated as a
     * sequence of 16-bit units (UTF-16 encoded characters). It is not a
     * requirement to have an XML declaration when using
     * <code>stringData</code>. If an XML declaration is present, the value
     * of the encoding attribute will be ignored.
     */
    public String getStringData();
    /**
     *  String data to parse. If provided, this will always be treated as a
     * sequence of 16-bit units (UTF-16 encoded characters). It is not a
     * requirement to have an XML declaration when using
     * <code>stringData</code>. If an XML declaration is present, the value
     * of the encoding attribute will be ignored.
     */
    public void setStringData(String stringData);

    /**
     *  The system identifier, a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>], for this
     * input source. The system identifier is optional if there is a byte
     * stream, a character stream, or string data. It is still useful to
     * provide one, since the application will use it to resolve any
     * relative URIs and can include it in error messages and warnings. (The
     * LSParser will only attempt to fetch the resource identified by the
```

```
     * URI reference if there is no other input available in the input
     * source.)
     * <br> If the application knows the character encoding of the object
     * pointed to by the system identifier, it can set the encoding using
     * the <code>encoding</code> attribute.
     * <br> If the specified system ID is a relative URI reference (see
     * section 5 in [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]), the DOM
     * implementation will attempt to resolve the relative URI with the
     * <code>baseURI</code> as the base, if that fails, the behavior is
     * implementation dependent.
     */
    public String getSystemId();
    /**
     *  The system identifier, a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF
         RFC 2396</a>], for this
     * input source. The system identifier is optional if there is a byte
     * stream, a character stream, or string data. It is still useful to
     * provide one, since the application will use it to resolve any
     * relative URIs and can include it in error messages and warnings. (The
     * LSParser will only attempt to fetch the resource identified by the
     * URI reference if there is no other input available in the input
     * source.)
     * <br> If the application knows the character encoding of the object
     * pointed to by the system identifier, it can set the encoding using
     * the <code>encoding</code> attribute.
     * <br> If the specified system ID is a relative URI reference (see
     * section 5 in [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]), the DOM
     * implementation will attempt to resolve the relative URI with the
     * <code>baseURI</code> as the base, if that fails, the behavior is
     * implementation dependent.
     */
    public void setSystemId(String systemId);

    /**
     *  The public identifier for this input source. This may be mapped to an
     * input source using an implementation dependent mechanism (such as
     * catalogues or other mappings). The public identifier, if specified,
     * may also be reported as part of the location information when errors
     * are reported.
     */
    public String getPublicId();
    /**
     *  The public identifier for this input source. This may be mapped to an
     * input source using an implementation dependent mechanism (such as
     * catalogues or other mappings). The public identifier, if specified,
     * may also be reported as part of the location information when errors
     * are reported.
     */
    public void setPublicId(String publicId);

    /**
     *  The base URI to be used (see section 5.1.4 in [<a href='http://www.ietf.org/rfc/rfc2396.txt
         '>IETF RFC 2396</a>]) for
```

```
     * resolving a relative <code>systemId</code> to an absolute URI.
     * <br> If, when used, the base URI is itself a relative URI, an empty
     * string, or null, the behavior is implementation dependent.
     */
    public String getBaseURI();
    /**
     *  The base URI to be used (see section 5.1.4 in [<a href='http://www.ietf.org/rfc/rfc2396.txt
         '>IETF RFC 2396</a>]) for
     * resolving a relative <code>systemId</code> to an absolute URI.
     * <br> If, when used, the base URI is itself a relative URI, an empty
     * string, or null, the behavior is implementation dependent.
     */
    public void setBaseURI(String baseURI);

    /**
     *  The character encoding, if known. The encoding must be a string
     * acceptable for an XML encoding declaration ([<a href='http://www.w3.org/TR/2004/REC-xml
         -20040204'>XML 1.0</a>] section
     * 4.3.3 "Character Encoding in Entities").
     * <br> This attribute has no effect when the application provides a
     * character stream or string data. For other sources of input, an
     * encoding specified by means of this attribute will override any
     * encoding specified in the XML declaration or the Text declaration, or
     * an encoding obtained from a higher level protocol, such as HTTP [<a href='http://www.ietf.
         org/rfc/rfc2616.txt'>IETF RFC 2616</a>].
     */
    public String getEncoding();
    /**
     *  The character encoding, if known. The encoding must be a string
     * acceptable for an XML encoding declaration ([<a href='http://www.w3.org/TR/2004/REC-xml
         -20040204'>XML 1.0</a>] section
     * 4.3.3 "Character Encoding in Entities").
     * <br> This attribute has no effect when the application provides a
     * character stream or string data. For other sources of input, an
     * encoding specified by means of this attribute will override any
     * encoding specified in the XML declaration or the Text declaration, or
     * an encoding obtained from a higher level protocol, such as HTTP [<a href='http://www.ietf.
         org/rfc/rfc2616.txt'>IETF RFC 2616</a>].
     */
    public void setEncoding(String encoding);

    /**
     *  If set to true, assume that the input is certified (see section 2.13
     * in [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>]) when
     * parsing [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>].
     */
    public boolean getCertifiedText();
    /**
     *  If set to true, assume that the input is certified (see section 2.13
     * in [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>]) when
     * parsing [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>].
     */
    public void setCertifiedText(boolean certifiedText);
```

```

}
```

41.4 LSLoadEvent.java

Secuencia 450: org/w3c/dom/ls/LSLoadEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

import org.w3c.dom.Document;
import org.w3c.dom.events.Event;

/**
```

```
 *  This interface represents a load event object that signals the completion
 * of a document load.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSLoadEvent extends Event {
    /**
     * The document that finished loading.
     */
    public Document getNewDocument();

    /**
     * The input source that was parsed.
     */
    public LSInput getInput();

}
```

41.5 LSOutput.java

Secuencia 451: org/w3c/dom/ls/LSOutput.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
```

```
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

/**
 *  This interface represents an output destination for data.
 * <p> This interface allows an application to encapsulate information about
 * an output destination in a single object, which may include a URI, a byte
 * stream (possibly with a specified encoding), a base URI, and/or a
 * character stream.
 * <p> The exact definitions of a byte stream and a character stream are
 * binding dependent.
 * <p> The application is expected to provide objects that implement this
 * interface whenever such objects are needed. The application can either
 * provide its own objects that implement this interface, or it can use the
 * generic factory method <code>DOMImplementationLS.createLSOutput()</code>
 * to create objects that implement this interface.
 * <p> The <code>LSSerializer</code> will use the <code>LSOutput</code> object
 * to determine where to serialize the output to. The
 * <code>LSSerializer</code> will look at the different outputs specified in
 * the <code>LSOutput</code> in the following order to know which one to
 * output to, the first one that is not null and not an empty string will be
 * used:
 * <ol>
 * <li> <code>LSOutput.characterStream</code>
 * </li>
 * <li>
 * <code>LSOutput.byteStream</code>
 * </li>
 * <li> <code>LSOutput.systemId</code>
 * </li>
 * </ol>
 * <p> <code>LSOutput</code> objects belong to the application. The DOM
 * implementation will never modify them (though it may make copies and
 * modify the copies, if necessary).
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSOutput {
    /**
     *  An attribute of a language and binding dependent type that represents
     * a writable stream to which 16-bit units can be output.
     */
    public java.io.Writer getCharacterStream();
```

```
    /**
     *  An attribute of a language and binding dependent type that represents
     * a writable stream to which 16-bit units can be output.
     */
    public void setCharacterStream(java.io.Writer characterStream);

    /**
     *  An attribute of a language and binding dependent type that represents
     * a writable stream of bytes.
     */
    public java.io.OutputStream getByteStream();
    /**
     *  An attribute of a language and binding dependent type that represents
     * a writable stream of bytes.
     */
    public void setByteStream(java.io.OutputStream byteStream);

    /**
     *  The system identifier, a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF
         RFC 2396</a>], for this
     * output destination.
     * <br> If the system ID is a relative URI reference (see section 5 in [<a href='http://www.
         ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]), the
     * behavior is implementation dependent.
     */
    public String getSystemId();
    /**
     *  The system identifier, a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF
         RFC 2396</a>], for this
     * output destination.
     * <br> If the system ID is a relative URI reference (see section 5 in [<a href='http://www.
         ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]), the
     * behavior is implementation dependent.
     */
    public void setSystemId(String systemId);

    /**
     *  The character encoding to use for the output. The encoding must be a
     * string acceptable for an XML encoding declaration ([<a href='http://www.w3.org/TR/2004/REC-
         xml-20040204'>XML 1.0</a>] section
     * 4.3.3 "Character Encoding in Entities"), it is recommended that
     * character encodings registered (as charsets) with the Internet
     * Assigned Numbers Authority [<a href='ftp://ftp.isi.edu/in-notes/iana/assignments/character-
         sets'>IANA-CHARSETS</a>]
     *  should be referred to using their registered names.
     */
    public String getEncoding();
    /**
     *  The character encoding to use for the output. The encoding must be a
     * string acceptable for an XML encoding declaration ([<a href='http://www.w3.org/TR/2004/REC-
         xml-20040204'>XML 1.0</a>] section
     * 4.3.3 "Character Encoding in Entities"), it is recommended that
     * character encodings registered (as charsets) with the Internet
```

```
     * Assigned Numbers Authority [<a href='ftp://ftp.isi.edu/in-notes/iana/assignments/character-
         sets'>IANA-CHARSETS</a>]
     *  should be referred to using their registered names.
     */
    public void setEncoding(String encoding);

}
```

41.6 LSParser.java

Secuencia 452: org/w3c/dom/ls/LSParser.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;
```

```java

import org.w3c.dom.Document;
import org.w3c.dom.DOMConfiguration;
import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 *  An interface to an object that is able to build, or augment, a DOM tree
 * from various input sources.
 * <p> <code>LSParser</code> provides an API for parsing XML and building the
 * corresponding DOM document structure. A <code>LSParser</code> instance
 * can be obtained by invoking the
 * <code>DOMImplementationLS.createLSParser()</code> method.
 * <p> As specified in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
 * , when a document is first made available via the LSParser:
 * <ul>
 * <li> there will
 * never be two adjacent nodes of type NODE_TEXT, and there will never be
 * empty text nodes.
 * </li>
 * <li> it is expected that the <code>value</code> and
 * <code>nodeValue</code> attributes of an <code>Attr</code> node initially
 * return the <a href='http://www.w3.org/TR/2004/REC-xml-20040204#AVNormalize'>XML 1.0
 * normalized value</a>. However, if the parameters "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-validate-if-schema'>
 * validate-if-schema</a>" and "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-datatype-normalization'>
 * datatype-normalization</a>" are set to <code>true</code>, depending on the attribute normalization
 * used, the attribute values may differ from the ones obtained by the XML
 * 1.0 attribute normalization. If the parameters "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-datatype-normalization'>
 * datatype-normalization</a>" is set to <code>false</code>, the XML 1.0 attribute normalization is
 * guaranteed to occur, and if the attributes list does not contain
 * namespace declarations, the <code>attributes</code> attribute on
 * <code>Element</code> node represents the property <b>[attributes]</b> defined in [<a href='http://www.w3.org/TR/2004/REC-xml-infoset-20040204/'>XML Information Set</a>]
 * .
 * </li>
 * </ul>
 * <p> Asynchronous <code>LSParser</code> objects are expected to also
 * implement the <code>events::EventTarget</code> interface so that event
 * listeners can be registered on asynchronous <code>LSParser</code>
 * objects.
 * <p> Events supported by asynchronous <code>LSParser</code> objects are:
 * <dl>
 * <dt>load</dt>
 * <dd>
 *  The <code>LSParser</code> finishes to load the document. See also the
 * definition of the <code>LSLoadEvent</code> interface. </dd>
 * <dt>progress</dt>
 * <dd> The
```

```
* <code>LSParser</code> signals progress as data is parsed.  This
* specification does not attempt to define exactly when progress events
* should be dispatched. That is intentionally left as
* implementation-dependent. Here is one example of how an application might
* dispatch progress events: Once the parser starts receiving data, a
* progress event is dispatched to indicate that the parsing starts. From
* there on, a progress event is dispatched for every 4096 bytes of data
* that is received and processed. This is only one example, though, and
* implementations can choose to dispatch progress events at any time while
* parsing, or not dispatch them at all.  See also the definition of the
* <code>LSProgressEvent</code> interface. </dd>
* </dl>
* <p ><b>Note:</b>  All events defined in this specification use the
* namespace URI <code>"http://www.w3.org/2002/DOMLS"</code>.
* <p> While parsing an input source, errors are reported to the application
* through the error handler (<code>LSParser.domConfig</code>'s "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-error-handler'>
* error-handler</a>" parameter). This specification does in no way try to define all possible
* errors that can occur while parsing XML, or any other markup, but some
* common error cases are defined. The types (<code>DOMError.type</code>) of
* errors and warnings defined by this specification are:
* <dl>
* <dt>
* <code>"check-character-normalization-failure" [error]</code> </dt>
* <dd> Raised if
* the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-check-character-normalization'>
* check-character-normalization</a>" is set to true and a string is encountered that fails normalization
* checking. </dd>
* <dt><code>"doctype-not-allowed" [fatal]</code></dt>
* <dd> Raised if the
* configuration parameter "disallow-doctype" is set to <code>true</code>
* and a doctype is encountered. </dd>
* <dt><code>"no-input-specified" [fatal]</code></dt>
* <dd>
* Raised when loading a document and no input is specified in the
* <code>LSInput</code> object. </dd>
* <dt>
* <code>"pi-base-uri-not-preserved" [warning]</code></dt>
* <dd> Raised if a processing
* instruction is encountered in a location where the base URI of the
* processing instruction can not be preserved.  One example of a case where
* this warning will be raised is if the configuration parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-entities'>
* entities</a>" is set to <code>false</code> and the following XML file is parsed:
* <pre>
* &lt;!DOCTYPE root [ &lt;!ENTITY e SYSTEM 'subdir/myentity.ent' ]&gt;
* &lt;root&gt; &amp;e; &lt;/root&gt;</pre>
*  And <code>subdir/myentity.ent</code>
* contains:
* <pre>&lt;one&gt; &lt;two/&gt; &lt;/one&gt; &lt;?pi 3.14159?&gt;
* &lt;more/&gt;</pre>
```

```
 * </dd>
 * <dt><code>"unbound-prefix-in-entity" [warning]</code></dt>
 * <dd> An
 * implementation dependent warning that may be raised if the configuration
 * parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-namespaces'>
 * namespaces</a>" is set to <code>true</code> and an unbound namespace prefix is
 * encountered in an entity's replacement text. Raising this warning is not
 * enforced since some existing parsers may not recognize unbound namespace
 * prefixes in the replacement text of entities. </dd>
 * <dt>
 * <code>"unknown-character-denormalization" [fatal]</code></dt>
 * <dd> Raised if the
 * configuration parameter "ignore-unknown-character-denormalizations" is
 * set to <code>false</code> and a character is encountered for which the
 * processor cannot determine the normalization properties. </dd>
 * <dt>
 * <code>"unsupported-encoding" [fatal]</code></dt>
 * <dd> Raised if an unsupported
 * encoding is encountered. </dd>
 * <dt><code>"unsupported-media-type" [fatal]</code></dt>
 * <dd>
 * Raised if the configuration parameter "supported-media-types-only" is set
 * to <code>true</code> and an unsupported media type is encountered. </dd>
 * </dl>
 * <p> In addition to raising the defined errors and warnings, implementations
 * are expected to raise implementation specific errors and warnings for any
 * other error and warning cases such as IO errors (file not found,
 * permission denied,...), XML well-formedness errors, and so on.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSParser {
    /**
     *  The <code>DOMConfiguration</code> object used when parsing an input
     * source. This <code>DOMConfiguration</code> is specific to the parse
     * operation. No parameter values from this <code>DOMConfiguration</code>
     *  object are passed automatically to the <code>DOMConfiguration</code>
     * object on the <code>Document</code> that is created, or used, by the
     * parse operation. The DOM application is responsible for passing any
     * needed parameter values from this <code>DOMConfiguration</code>
     * object to the <code>DOMConfiguration</code> object referenced by the
     * <code>Document</code> object.
     * <br> In addition to the parameters recognized in on the <a href='http://www.w3.org/TR/DOM-
         Level-3-Core/core.html#DOMConfiguration'>
     * DOMConfiguration</a> interface defined in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level
         -3-Core-20040407'>DOM Level 3 Core</a>]
     * , the <code>DOMConfiguration</code> objects for <code>LSParser</code>
     * add or modify the following parameters:
     * <dl>
     * <dt>
     * <code>"charset-overrides-xml-encoding"</code></dt>
     * <dd>
```

```
     * <dl>
     * <dt><code>true</code></dt>
     * <dd>[<em>optional</em>] (<em>default</em>) If a higher level protocol such as HTTP [<a href
         ='http://www.ietf.org/rfc/rfc2616.txt'>IETF RFC 2616</a>] provides an
     * indication of the character encoding of the input stream being
     * processed, that will override any encoding specified in the XML
     * declaration or the Text declaration (see also section 4.3.3,
     * "Character Encoding in Entities", in [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>
         XML 1.0</a>]).
     * Explicitly setting an encoding in the <code>LSInput</code> overrides
     * any encoding from the protocol. </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>required</em>] The parser ignores any character set encoding information from
     * higher-level protocols. </dd>
     * </dl></dd>
     * <dt><code>"disallow-doctype"</code></dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>optional</em>] Throw a fatal <b>"doctype-not-allowed"</b> error if a doctype node
         is found while parsing the document. This is
     * useful when dealing with things like SOAP envelopes where doctype
     * nodes are not allowed. </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Allow doctype nodes in the document. </dd>
     * </dl></dd>
     * <dt>
     * <code>"ignore-unknown-character-denormalizations"</code></dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) If, while verifying full normalization when [<a
         href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>] is
     * supported, a processor encounters characters for which it cannot
     * determine the normalization properties, then the processor will
     * ignore any possible denormalizations caused by these characters.
     * This parameter is ignored for [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML
         1.0</a>]. </dd>
     * <dt>
     * <code>false</code></dt>
     * <dd>[<em>optional</em>] Report an fatal <b>"unknown-character-denormalization"</b> error if
         a character is encountered for which the processor cannot
     * determine the normalization properties. </dd>
     * </dl></dd>
     * <dt><code>"infoset"</code></dt>
     * <dd> See
     * the definition of <code>DOMConfiguration</code> for a description of
     * this parameter. Unlike in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core
         -20040407'>DOM Level 3 Core</a>]
     * , this parameter will default to <code>true</code> for
     * <code>LSParser</code>. </dd>
```

```
     * <dt><code>"namespaces"</code></dt>
     * <dd>
     * <dl>
     * <dt><code>true</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Perform the namespace processing as defined in [<a href='http://www.w3.org/TR/1999/REC-xml-names-19990114/'>XML Namespaces</a>]
     *  and [<a href='http://www.w3.org/TR/2004/REC-xml-names11-20040204/'>XML Namespaces 1.1</a>]
     * . </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>optional</em>] Do not perform the namespace processing. </dd>
     * </dl></dd>
     * <dt>
     * <code>"resource-resolver"</code></dt>
     * <dd>[<em>required</em>] A reference to a <code>LSResourceResolver</code> object, or null. If
     * the value of this parameter is not null when an external resource
     * (such as an external XML entity or an XML schema location) is
     * encountered, the implementation will request that the
     * <code>LSResourceResolver</code> referenced in this parameter resolves
     * the resource. </dd>
     * <dt><code>"supported-media-types-only"</code></dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>optional</em>] Check that the media type of the parsed resource is a supported media
     * type. If an unsupported media type is encountered, a fatal error of
     * type <b>"unsupported-media-type"</b> will be raised. The media types defined in [<a href='http://www.ietf.org/rfc/rfc3023.txt'>IETF RFC 3023</a>] must always
     * be accepted. </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Accept any media type. </dd>
     * </dl></dd>
     * <dt><code>"validate"</code></dt>
     * <dd> See the definition of
     * <code>DOMConfiguration</code> for a description of this parameter.
     * Unlike in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     * , the processing of the internal subset is always accomplished, even
     * if this parameter is set to <code>false</code>. </dd>
     * <dt>
     * <code>"validate-if-schema"</code></dt>
     * <dd> See the definition of
     * <code>DOMConfiguration</code> for a description of this parameter.
     * Unlike in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     * , the processing of the internal subset is always accomplished, even
     * if this parameter is set to <code>false</code>. </dd>
     * <dt>
     * <code>"well-formed"</code></dt>
     * <dd> See the definition of
     * <code>DOMConfiguration</code> for a description of this parameter.
     * Unlike in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3
```

```
        Core</a>]
     * , this parameter cannot be set to <code>false</code>. </dd>
     * </dl>
     */
    public DOMConfiguration getDomConfig();

    /**
     *  When a filter is provided, the implementation will call out to the
     * filter as it is constructing the DOM tree structure. The filter can
     * choose to remove elements from the document being constructed, or to
     * terminate the parsing early.
     * <br> The filter is invoked after the operations requested by the
     * <code>DOMConfiguration</code> parameters have been applied. For
     * example, if "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-validate'>
     * validate</a>" is set to <code>true</code>, the validation is done before invoking the
     * filter.
     */
    public LSParserFilter getFilter();
    /**
     *  When a filter is provided, the implementation will call out to the
     * filter as it is constructing the DOM tree structure. The filter can
     * choose to remove elements from the document being constructed, or to
     * terminate the parsing early.
     * <br> The filter is invoked after the operations requested by the
     * <code>DOMConfiguration</code> parameters have been applied. For
     * example, if "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-validate'>
     * validate</a>" is set to <code>true</code>, the validation is done before invoking the
     * filter.
     */
    public void setFilter(LSParserFilter filter);

    /**
     *  <code>true</code> if the <code>LSParser</code> is asynchronous,
     * <code>false</code> if it is synchronous.
     */
    public boolean getAsync();

    /**
     *  <code>true</code> if the <code>LSParser</code> is currently busy
     * loading a document, otherwise <code>false</code>.
     */
    public boolean getBusy();

    /**
     * Parse an XML document from a resource identified by a
     * <code>LSInput</code>.
     * @param input  The <code>LSInput</code> from which the source of the
     *   document is to be read.
     * @return  If the <code>LSParser</code> is a synchronous
     *   <code>LSParser</code>, the newly created and populated
     *   <code>Document</code> is returned. If the <code>LSParser</code> is
     *   asynchronous, <code>null</code> is returned since the document
     *   object may not yet be constructed when this method returns.
```

```
     * @exception DOMException
     *    INVALID_STATE_ERR: Raised if the <code>LSParser</code>'s
     *   <code>LSParser.busy</code> attribute is <code>true</code>.
     * @exception LSException
     *    PARSE_ERR: Raised if the <code>LSParser</code> was unable to load
     *   the XML document. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
    public Document parse(LSInput input)
                          throws DOMException, LSException;

    /**
     *  Parse an XML document from a location identified by a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]. If the URI
     * contains a fragment identifier (see section 4.1 in [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>]), the
     * behavior is not defined by this specification, future versions of
     * this specification may define the behavior.
     * @param uri The location of the XML document to be read.
     * @return  If the <code>LSParser</code> is a synchronous
     *   <code>LSParser</code>, the newly created and populated
     *   <code>Document</code> is returned, or <code>null</code> if an error
     *   occured. If the <code>LSParser</code> is asynchronous,
     *   <code>null</code> is returned since the document object may not yet
     *   be constructed when this method returns.
     * @exception DOMException
     *    INVALID_STATE_ERR: Raised if the <code>LSParser.busy</code>
     *   attribute is <code>true</code>.
     * @exception LSException
     *    PARSE_ERR: Raised if the <code>LSParser</code> was unable to load
     *   the XML document. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
    public Document parseURI(String uri)
                            throws DOMException, LSException;

    // ACTION_TYPES
    /**
     *  Append the result of the parse operation as children of the context
     * node. For this action to work, the context node must be an
     * <code>Element</code> or a <code>DocumentFragment</code>.
     */
    public static final short ACTION_APPEND_AS_CHILDREN = 1;
    /**
     *  Replace all the children of the context node with the result of the
     * parse operation. For this action to work, the context node must be an
     * <code>Element</code>, a <code>Document</code>, or a
     * <code>DocumentFragment</code>.
     */
```

```
    public static final short ACTION_REPLACE_CHILDREN   = 2;
    /**
     *  Insert the result of the parse operation as the immediately preceding
     * sibling of the context node. For this action to work the context
     * node's parent must be an <code>Element</code> or a
     * <code>DocumentFragment</code>.
     */
    public static final short ACTION_INSERT_BEFORE      = 3;
    /**
     *  Insert the result of the parse operation as the immediately following
     * sibling of the context node. For this action to work the context
     * node's parent must be an <code>Element</code> or a
     * <code>DocumentFragment</code>.
     */
    public static final short ACTION_INSERT_AFTER       = 4;
    /**
     *  Replace the context node with the result of the parse operation. For
     * this action to work, the context node must have a parent, and the
     * parent must be an <code>Element</code> or a
     * <code>DocumentFragment</code>.
     */
    public static final short ACTION_REPLACE            = 5;

    /**
     *  Parse an XML fragment from a resource identified by a
     * <code>LSInput</code> and insert the content into an existing document
     * at the position specified with the <code>context</code> and
     * <code>action</code> arguments. When parsing the input stream, the
     * context node (or its parent, depending on where the result will be
     * inserted) is used for resolving unbound namespace prefixes. The
     * context node's <code>ownerDocument</code> node (or the node itself if
     * the node of type <code>DOCUMENT_NODE</code>) is used to resolve
     * default attributes and entity references.
     * <br> As the new data is inserted into the document, at least one
     * mutation event is fired per new immediate child or sibling of the
     * context node.
     * <br> If the context node is a <code>Document</code> node and the action
     * is <code>ACTION_REPLACE_CHILDREN</code>, then the document that is
     * passed as the context node will be changed such that its
     * <code>xmlEncoding</code>, <code>documentURI</code>,
     * <code>xmlVersion</code>, <code>inputEncoding</code>,
     * <code>xmlStandalone</code>, and all other such attributes are set to
     * what they would be set to if the input source was parsed using
     * <code>LSParser.parse()</code>.
     * <br> This method is always synchronous, even if the
     * <code>LSParser</code> is asynchronous (<code>LSParser.async</code> is
     * <code>true</code>).
     * <br> If an error occurs while parsing, the caller is notified through
     * the <code>ErrorHandler</code> instance associated with the "<a href='http://www.w3.org/TR/
         DOM-Level-3-Core/core.html#parameter-error-handler'>
     * error-handler</a>" parameter of the <code>DOMConfiguration</code>.
     * <br> When calling <code>parseWithContext</code>, the values of the
     * following configuration parameters will be ignored and their default
```

```
     * values will always be used instead: "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.
         html#parameter-validate'>
     * validate</a>", "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-validate-
         if-schema'>
     * validate-if-schema</a>", and "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#
         parameter-element-content-whitespace'>
     * element-content-whitespace</a>". Other parameters will be treated normally, and the parser
         is expected
     * to call the <code>LSParserFilter</code> just as if a whole document
     * was parsed.
     * @param input  The <code>LSInput</code> from which the source document
     *   is to be read. The source document must be an XML fragment, i.e.
     *   anything except a complete XML document (except in the case where
     *   the context node of type <code>DOCUMENT_NODE</code>, and the action
     *   is <code>ACTION_REPLACE_CHILDREN</code>), a DOCTYPE (internal
     *   subset), entity declaration(s), notation declaration(s), or XML or
     *   text declaration(s).
     * @param contextArg  The node that is used as the context for the data
     *   that is being parsed. This node must be a <code>Document</code>
     *   node, a <code>DocumentFragment</code> node, or a node of a type
     *   that is allowed as a child of an <code>Element</code> node, e.g. it
     *   cannot be an <code>Attribute</code> node.
     * @param action  This parameter describes which action should be taken
     *   between the new set of nodes being inserted and the existing
     *   children of the context node. The set of possible actions is
     *   defined in <code>ACTION_TYPES</code> above.
     * @return  Return the node that is the result of the parse operation. If
     *   the result is more than one top-level node, the first one is
     *   returned.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if the content cannot replace, be
     *   inserted before, after, or as a child of the context node (see also
     *   <code>Node.insertBefore</code> or <code>Node.replaceChild</code> in [<a href='http://www.
         w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     *   ).
     *   <br> NOT_SUPPORTED_ERR: Raised if the <code>LSParser</code> doesn't
     *   support this method, or if the context node is of type
     *   <code>Document</code> and the DOM implementation doesn't support
     *   the replacement of the <code>DocumentType</code> child or
     *   <code>Element</code> child.
     *   <br> NO_MODIFICATION_ALLOWED_ERR: Raised if the context node is a
     *   read only node and the content is being appended to its child list,
     *   or if the parent node of the context node is read only node and the
     *   content is being inserted in its child list.
     *   <br> INVALID_STATE_ERR: Raised if the <code>LSParser.busy</code>
     *   attribute is <code>true</code>.
     * @exception LSException
     *    PARSE_ERR: Raised if the <code>LSParser</code> was unable to load
     *   the XML fragment. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level
         -3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
```

```
    public Node parseWithContext(LSInput input,
                                 Node contextArg,
                                 short action)
                                 throws DOMException, LSException;

    /**
     *  Abort the loading of the document that is currently being loaded by
     * the <code>LSParser</code>. If the <code>LSParser</code> is currently
     * not busy, a call to this method does nothing.
     */
    public void abort();

}
```

41.7 LSParserFilter.java

Secuencia 453: org/w3c/dom/ls/LSParserFilter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
```

```
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

import org.w3c.dom.Node;
import org.w3c.dom.Element;

/**
 *  <code>LSParserFilter</code>s provide applications the ability to examine
 * nodes as they are being constructed while parsing. As each node is
 * examined, it may be modified or removed, or the entire parse may be
 * terminated early.
 * <p> At the time any of the filter methods are called by the parser, the
 * owner Document and DOMImplementation objects exist and are accessible.
 * The document element is never passed to the <code>LSParserFilter</code>
 * methods, i.e. it is not possible to filter out the document element.
 * <code>Document</code>, <code>DocumentType</code>, <code>Notation</code>,
 * <code>Entity</code>, and <code>Attr</code> nodes are never passed to the
 * <code>acceptNode</code> method on the filter. The child nodes of an
 * <code>EntityReference</code> node are passed to the filter if the
 * parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-entities'>
 * entities</a>" is set to <code>false</code>. Note that, as described by the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-entities'>
 * entities</a>", unexpanded entity reference nodes are never discarded and are always
 * passed to the filter.
 * <p> All validity checking while parsing a document occurs on the source
 * document as it appears on the input stream, not on the DOM document as it
 * is built in memory. With filters, the document in memory may be a subset
 * of the document on the stream, and its validity may have been affected by
 * the filtering.
 * <p> All default attributes must be present on elements when the elements
 * are passed to the filter methods. All other default content must be
 * passed to the filter methods.
 * <p> DOM applications must not raise exceptions in a filter. The effect of
 * throwing exceptions from a filter is DOM implementation dependent.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSParserFilter {
    // Constants returned by startElement and acceptNode
    /**
     * Accept the node.
     */
    public static final short FILTER_ACCEPT             = 1;
    /**
     * Reject the node and its children.
     */
    public static final short FILTER_REJECT             = 2;
    /**
```

```
     * Skip this single node. The children of this node will still be
     * considered.
     */
    public static final short FILTER_SKIP               = 3;
    /**
     *  Interrupt the normal processing of the document.
     */
    public static final short FILTER_INTERRUPT          = 4;

    /**
     *  The parser will call this method after each <code>Element</code> start
     * tag has been scanned, but before the remainder of the
     * <code>Element</code> is processed. The intent is to allow the
     * element, including any children, to be efficiently skipped. Note that
     * only element nodes are passed to the <code>startElement</code>
     * function.
     * <br>The element node passed to <code>startElement</code> for filtering
     * will include all of the Element's attributes, but none of the
     * children nodes. The Element may not yet be in place in the document
     * being constructed (it may not have a parent node.)
     * <br>A <code>startElement</code> filter function may access or change
     * the attributes for the Element. Changing Namespace declarations will
     * have no effect on namespace resolution by the parser.
     * <br>For efficiency, the Element node passed to the filter may not be
     * the same one as is actually placed in the tree if the node is
     * accepted. And the actual node (node object identity) may be reused
     * during the process of reading in and filtering a document.
     * @param elementArg The newly encountered element. At the time this
     *   method is called, the element is incomplete - it will have its
     *   attributes, but no children.
     * @return
     * <ul>
     * <li> <code>FILTER_ACCEPT</code> if the <code>Element</code> should
     *   be included in the DOM document being built.
     * </li>
     * <li>
     *   <code>FILTER_REJECT</code> if the <code>Element</code> and all of
     *   its children should be rejected.
     * </li>
     * <li> <code>FILTER_SKIP</code> if the
     *   <code>Element</code> should be skipped. All of its children are
     *   inserted in place of the skipped <code>Element</code> node.
     * </li>
     * <li>
     *   <code>FILTER_INTERRUPT</code> if the filter wants to stop the
     *   processing of the document. Interrupting the processing of the
     *   document does no longer guarantee that the resulting DOM tree is
     *   XML well-formed. The <code>Element</code> is rejected.
     * </li>
     * </ul> Returning
     *   any other values will result in unspecified behavior.
     */
    public short startElement(Element elementArg);
```

```

    /**
     * This method will be called by the parser at the completion of the
     * parsing of each node. The node and all of its descendants will exist
     * and be complete. The parent node will also exist, although it may be
     * incomplete, i.e. it may have additional children that have not yet
     * been parsed. Attribute nodes are never passed to this function.
     * <br>From within this method, the new node may be freely modified -
     * children may be added or removed, text nodes modified, etc. The state
     * of the rest of the document outside this node is not defined, and the
     * affect of any attempt to navigate to, or to modify any other part of
     * the document is undefined.
     * <br>For validating parsers, the checks are made on the original
     * document, before any modification by the filter. No validity checks
     * are made on any document modifications made by the filter.
     * <br>If this new node is rejected, the parser might reuse the new node
     * and any of its descendants.
     * @param nodeArg The newly constructed element. At the time this method
     *   is called, the element is complete - it has all of its children
     *   (and their children, recursively) and attributes, and is attached
     *   as a child to its parent.
     * @return
     * <ul>
     * <li> <code>FILTER_ACCEPT</code> if this <code>Node</code> should
     *   be included in the DOM document being built.
     * </li>
     * <li>
     *   <code>FILTER_REJECT</code> if the <code>Node</code> and all of its
     *   children should be rejected.
     * </li>
     * <li> <code>FILTER_SKIP</code> if the
     *   <code>Node</code> should be skipped and the <code>Node</code>
     *   should be replaced by all the children of the <code>Node</code>.
     * </li>
     * <li>
     *   <code>FILTER_INTERRUPT</code> if the filter wants to stop the
     *   processing of the document. Interrupting the processing of the
     *   document does no longer guarantee that the resulting DOM tree is
     *   XML well-formed. The <code>Node</code> is accepted and will be the
     *   last completely parsed node.
     * </li>
     * </ul>
     */
    public short acceptNode(Node nodeArg);

    /**
     *  Tells the <code>LSParser</code> what types of nodes to show to the
     * method <code>LSParserFilter.acceptNode</code>. If a node is not shown
     * to the filter using this attribute, it is automatically included in
     * the DOM document being built. See <code>NodeFilter</code> for
     * definition of the constants. The constants <code>SHOW_ATTRIBUTE</code>
     * , <code>SHOW_DOCUMENT</code>, <code>SHOW_DOCUMENT_TYPE</code>,
     * <code>SHOW_NOTATION</code>, <code>SHOW_ENTITY</code>, and
```

```
     * <code>SHOW_DOCUMENT_FRAGMENT</code> are meaningless here. Those nodes
     * will never be passed to <code>LSParserFilter.acceptNode</code>.
     * <br> The constants used here are defined in [<a href='http://www.w3.org/TR/2000/REC-DOM-
         Level-2-Traversal-Range-20001113'>DOM Level 2 Traversal and      Range</a>]
     * .
     */
    public int getWhatToShow();

}
```

41.8 LSProgressEvent.java

Secuencia 454: org/w3c/dom/ls/LSProgressEvent.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */
```

```

package org.w3c.dom.ls;

import org.w3c.dom.events.Event;

/**
 *  This interface represents a progress event object that notifies the
 * application about progress as a document is parsed. It extends the
 * <code>Event</code> interface defined in [<a href='http://www.w3.org/TR/2003/NOTE-DOM-Level-3-Events-20031107'>DOM Level 3 Events</a>]
 * .
 * <p> The units used for the attributes <code>position</code> and
 * <code>totalSize</code> are not specified and can be implementation and
 * input dependent.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSProgressEvent extends Event {
    /**
     * The input source that is being parsed.
     */
    public LSInput getInput();

    /**
     * The current position in the input source, including all external
     * entities and other resources that have been read.
     */
    public int getPosition();

    /**
     * The total size of the document including all external resources, this
     * number might change as a document is being parsed if references to
     * more external resources are seen. A value of <code>0</code> is
     * returned if the total size cannot be determined or estimated.
     */
    public int getTotalSize();

}
```

41.9 LSResourceResolver.java

Secuencia 455: org/w3c/dom/ls/LSResourceResolver.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

/**
 *  <code>LSResourceResolver</code> provides a way for applications to
 * redirect references to external resources.
 * <p> Applications needing to implement custom handling for external
 * resources can implement this interface and register their implementation
 * by setting the "resource-resolver" parameter of
 * <code>DOMConfiguration</code> objects attached to <code>LSParser</code>
 * and <code>LSSerializer</code>. It can also be register on
 * <code>DOMConfiguration</code> objects attached to <code>Document</code>
 * if the "LS" feature is supported.
 * <p> The <code>LSParser</code> will then allow the application to intercept
 * any external entities, including the external DTD subset and external
 * parameter entities, before including them. The top-level document entity
 * is never passed to the <code>resolveResource</code> method.
 * <p> Many DOM applications will not need to implement this interface, but it
 * will be especially useful for applications that build XML documents from
 * databases or other specialized input sources, or for applications that
 * use URNs.
 * <p ><b>Note:</b>  <code>LSResourceResolver</code> is based on the SAX2 [<a href='http://www.
     saxproject.org/'>SAX</a>] <code>EntityResolver</code>
```

```
 * interface.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSResourceResolver {
    /**
     *  Allow the application to resolve external resources.
     * <br> The <code>LSParser</code> will call this method before opening any
     * external resource, including the external DTD subset, external
     * entities referenced within the DTD, and external entities referenced
     * within the document element (however, the top-level document entity
     * is not passed to this method). The application may then request that
     * the <code>LSParser</code> resolve the external resource itself, that
     * it use an alternative URI, or that it use an entirely different input
     * source.
     * <br> Application writers can use this method to redirect external
     * system identifiers to secure and/or local URI, to look up public
     * identifiers in a catalogue, or to read an entity from a database or
     * other input source (including, for example, a dialog box).
     * @param type  The type of the resource being resolved. For XML [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a>] resources
     *   (i.e. entities), applications must use the value
     *   <code>"http://www.w3.org/TR/REC-xml"</code>. For XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
     *   , applications must use the value
     *   <code>"http://www.w3.org/2001/XMLSchema"</code>. Other types of
     *   resources are outside the scope of this specification and therefore
     *   should recommend an absolute URI in order to use this method.
     * @param namespaceURI  The namespace of the resource being resolved,
     *   e.g. the target namespace of the XML Schema [<a href='http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/'>XML Schema Part 1</a>]
     *    when resolving XML Schema resources.
     * @param publicId  The public identifier of the external entity being
     *   referenced, or <code>null</code> if no public identifier was
     *   supplied or if the resource is not an entity.
     * @param systemId  The system identifier, a URI reference [<a href='http://www.ietf.org/rfc/rfc2396.txt'>IETF RFC 2396</a>], of the
     *   external resource being referenced, or <code>null</code> if no
     *   system identifier was supplied.
     * @param baseURI  The absolute base URI of the resource being parsed, or
     *   <code>null</code> if there is no base URI.
     * @return  A <code>LSInput</code> object describing the new input
     *   source, or <code>null</code> to request that the parser open a
     *   regular URI connection to the resource.
     */
    public LSInput resolveResource(String type,
                                   String namespaceURI,
                                   String publicId,
                                   String systemId,
                                   String baseURI);

}
```

41.10 LSSerializer.java

Secuencia 456: org/w3c/dom/ls/LSSerializer.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

import org.w3c.dom.DOMConfiguration;
import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 *  A <code>LSSerializer</code> provides an API for serializing (writing) a
 * DOM document out into XML. The XML data is written to a string or an
```

```
* output stream. Any changes or fixups made during the serialization affect
* only the serialized data. The <code>Document</code> object and its
* children are never altered by the serialization operation.
* <p> During serialization of XML data, namespace fixup is done as defined in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
* , Appendix B. [<a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113'>DOM Level 2 Core</a>]
*  allows empty strings as a real namespace URI. If the
* <code>namespaceURI</code> of a <code>Node</code> is empty string, the
* serialization will treat them as <code>null</code>, ignoring the prefix
* if any.
* <p> <code>LSSerializer</code> accepts any node type for serialization. For
* nodes of type <code>Document</code> or <code>Entity</code>, well-formed
* XML will be created when possible (well-formedness is guaranteed if the
* document or entity comes from a parse operation and is unchanged since it
* was created). The serialized output for these node types is either as a
* XML document or an External XML Entity, respectively, and is acceptable
* input for an XML parser. For all other types of nodes the serialized form
* is implementation dependent.
* <p>Within a <code>Document</code>, <code>DocumentFragment</code>, or
* <code>Entity</code> being serialized, <code>Nodes</code> are processed as
* follows
* <ul>
* <li> <code>Document</code> nodes are written, including the XML
* declaration (unless the parameter "xml-declaration" is set to
* <code>false</code>) and a DTD subset, if one exists in the DOM. Writing a
* <code>Document</code> node serializes the entire document.
* </li>
* <li>
* <code>Entity</code> nodes, when written directly by
* <code>LSSerializer.write</code>, outputs the entity expansion but no
* namespace fixup is done. The resulting output will be valid as an
* external entity.
* </li>
* <li> If the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-entities'>
* entities</a>" is set to <code>true</code>, <code>EntityReference</code> nodes are
* serialized as an entity reference of the form "
* <code>&amp;entityName;</code>" in the output. Child nodes (the expansion)
* of the entity reference are ignored. If the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-entities'>
* entities</a>" is set to <code>false</code>, only the children of the entity reference
* are serialized. <code>EntityReference</code> nodes with no children (no
* corresponding <code>Entity</code> node or the corresponding
* <code>Entity</code> nodes have no children) are always serialized.
* </li>
* <li>
* <code>CDATAsections</code> containing content characters that cannot be
* represented in the specified output encoding are handled according to the
* "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-split-cdata-sections'>
* split-cdata-sections</a>" parameter.  If the parameter is set to <code>true</code>,
* <code>CDATAsections</code> are split, and the unrepresentable characters
* are serialized as numeric character references in ordinary content. The
```

```
* exact position and number of splits is not specified.  If the parameter
* is set to <code>false</code>, unrepresentable characters in a
* <code>CDATAsection</code> are reported as
* <code>"wf-invalid-character"</code> errors if the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-well-formed'>
* well-formed</a>" is set to <code>true</code>. The error is not recoverable - there is no
* mechanism for supplying alternative characters and continuing with the
* serialization.
* </li>
* <li> <code>DocumentFragment</code> nodes are serialized by
* serializing the children of the document fragment in the order they
* appear in the document fragment.
* </li>
* <li> All other node types (Element, Text,
* etc.) are serialized to their corresponding XML source form.
* </li>
* </ul>
* <p ><b>Note:</b>  The serialization of a <code>Node</code> does not always
* generate a well-formed XML document, i.e. a <code>LSParser</code> might
* throw fatal errors when parsing the resulting serialization.
* <p> Within the character data of a document (outside of markup), any
* characters that cannot be represented directly are replaced with
* character references. Occurrences of '&lt;' and '&amp;' are replaced by
* the predefined entities &amp;lt; and &amp;amp;. The other predefined
* entities (&amp;gt;, &amp;apos;, and &amp;quot;) might not be used, except
* where needed (e.g. using &amp;gt; in cases such as ']]&gt;'). Any
* characters that cannot be represented directly in the output character
* encoding are serialized as numeric character references (and since
* character encoding standards commonly use hexadecimal representations of
* characters, using the hexadecimal representation when serializing
* character references is encouraged).
* <p> To allow attribute values to contain both single and double quotes, the
* apostrophe or single-quote character (') may be represented as
* "&amp;apos;", and the double-quote character (")  as "&amp;quot;". New
* line characters and other characters that cannot be represented directly
* in attribute values in the output character encoding are serialized as a
* numeric character reference.
* <p> Within markup, but outside of attributes, any occurrence of a character
* that cannot be represented in the output character encoding is reported
* as a <code>DOMError</code> fatal error. An example would be serializing
* the element &lt;LaCa\u00f1ada/&gt; with <code>encoding="us-ascii"</code>.
* This will result with a generation of a <code>DOMError</code>
* "wf-invalid-character-in-node-name" (as proposed in "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-well-formed'>
* well-formed</a>").
* <p> When requested by setting the parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-normalize-characters'>
* normalize-characters</a>" on <code>LSSerializer</code> to true, character normalization is
* performed according to the definition of <a href='http://www.w3.org/TR/2004/REC-xml11-20040204/#dt-fullnorm'>fully
* normalized</a> characters included in appendix E of [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>] on all
* data to be serialized, both markup and character data. The character
```

```
* normalization process affects only the data as it is being written; it
* does not alter the DOM's view of the document after serialization has
* completed.
* <p> Implementations are required to support the encodings "UTF-8",
* "UTF-16", "UTF-16BE", and "UTF-16LE" to guarantee that data is
* serializable in all encodings that are required to be supported by all
* XML parsers. When the encoding is UTF-8, whether or not a byte order mark
* is serialized, or if the output is big-endian or little-endian, is
* implementation dependent. When the encoding is UTF-16, whether or not the
* output is big-endian or little-endian is implementation dependent, but a
* Byte Order Mark must be generated for non-character outputs, such as
* <code>LSOutput.byteStream</code> or <code>LSOutput.systemId</code>. If
* the Byte Order Mark is not generated, a "byte-order-mark-needed" warning
* is reported. When the encoding is UTF-16LE or UTF-16BE, the output is
* big-endian (UTF-16BE) or little-endian (UTF-16LE) and the Byte Order Mark
* is not be generated. In all cases, the encoding declaration, if
* generated, will correspond to the encoding used during the serialization
* (e.g. <code>encoding="UTF-16"</code> will appear if UTF-16 was
* requested).
* <p> Namespaces are fixed up during serialization, the serialization process
* will verify that namespace declarations, namespace prefixes and the
* namespace URI associated with elements and attributes are consistent. If
* inconsistencies are found, the serialized form of the document will be
* altered to remove them. The method used for doing the namespace fixup
* while serializing a document is the algorithm defined in Appendix B.1,
* "Namespace normalization", of [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core
    -20040407'>DOM Level 3 Core</a>]
* .
* <p> While serializing a document, the parameter "discard-default-content"
* controls whether or not non-specified data is serialized.
* <p> While serializing, errors and warnings are reported to the application
* through the error handler (<code>LSSerializer.domConfig</code>'s "<a href='http://www.w3.org/TR/
    DOM-Level-3-Core/core.html#parameter-error-handler'>
* error-handler</a>" parameter). This specification does in no way try to define all possible
* errors and warnings that can occur while serializing a DOM node, but some
* common error and warning cases are defined. The types (
* <code>DOMError.type</code>) of errors and warnings defined by this
* specification are:
* <dl>
* <dt><code>"no-output-specified" [fatal]</code></dt>
* <dd> Raised when
* writing to a <code>LSOutput</code> if no output is specified in the
* <code>LSOutput</code>. </dd>
* <dt>
* <code>"unbound-prefix-in-entity-reference" [fatal]</code> </dt>
* <dd> Raised if the
* configuration parameter "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-
    namespaces'>
* namespaces</a>" is set to <code>true</code> and an entity whose replacement text
* contains unbound namespace prefixes is referenced in a location where
* there are no bindings for the namespace prefixes. </dd>
* <dt>
* <code>"unsupported-encoding" [fatal]</code></dt>
```

```
 * <dd> Raised if an unsupported
 * encoding is encountered. </dd>
 * </dl>
 * <p> In addition to raising the defined errors and warnings, implementations
 * are expected to raise implementation specific errors and warnings for any
 * other error and warning cases such as IO errors (file not found,
 * permission denied,...) and so on.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSSerializer {
    /**
     *  The <code>DOMConfiguration</code> object used by the
     * <code>LSSerializer</code> when serializing a DOM node.
     * <br> In addition to the parameters recognized by the <a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#DOMConfiguration'>
     * DOMConfiguration</a> interface defined in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     * , the <code>DOMConfiguration</code> objects for
     * <code>LSSerializer</code> adds, or modifies, the following
     * parameters:
     * <dl>
     * <dt><code>"canonical-form"</code></dt>
     * <dd>
     * <dl>
     * <dt><code>true</code></dt>
     * <dd>[<em>optional</em>] Writes the document according to the rules specified in [<a href='http://www.w3.org/TR/2001/REC-xml-c14n-20010315'>Canonical XML</a>].
     * In addition to the behavior described in "<a href='http://www.w3.org/TR/DOM-Level-3-Core/core.html#parameter-canonical-form'>
     * canonical-form</a>" [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     * , setting this parameter to <code>true</code> will set the parameters
     * "format-pretty-print", "discard-default-content", and "xml-declaration
     * ", to <code>false</code>. Setting one of those parameters to
     * <code>true</code> will set this parameter to <code>false</code>.
     * Serializing an XML 1.1 document when "canonical-form" is
     * <code>true</code> will generate a fatal error. </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Do not canonicalize the output. </dd>
     * </dl></dd>
     * <dt><code>"discard-default-content"</code></dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Use the <code>Attr.specified</code> attribute to decide what attributes
     * should be discarded. Note that some implementations might use
     * whatever information available to the implementation (i.e. XML
     * schema, DTD, the <code>Attr.specified</code> attribute, and so on) to
     * determine what attributes and content to discard if this parameter is
```

```
     * set to <code>true</code>. </dd>
     * <dt><code>false</code></dt>
     * <dd>[<em>required</em>]Keep all attributes and all content.</dd>
     * </dl></dd>
     * <dt><code>"format-pretty-print"</code></dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>optional</em>] Formatting the output by adding whitespace to produce a pretty-printed,
     * indented, human-readable form. The exact form of the transformations
     * is not specified by this specification. Pretty-printing changes the
     * content of the document and may affect the validity of the document,
     * validating implementations should preserve validity. </dd>
     * <dt>
     * <code>false</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) Don't pretty-print the result. </dd>
     * </dl></dd>
     * <dt>
     * <code>"ignore-unknown-character-denormalizations"</code> </dt>
     * <dd>
     * <dl>
     * <dt>
     * <code>true</code></dt>
     * <dd>[<em>required</em>] (<em>default</em>) If, while verifying full normalization when [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>] is
     * supported, a character is encountered for which the normalization
     * properties cannot be determined, then raise a
     * <code>"unknown-character-denormalization"</code> warning (instead of
     * raising an error, if this parameter is not set) and ignore any
     * possible denormalizations caused by these characters. </dd>
     * <dt>
     * <code>false</code></dt>
     * <dd>[<em>optional</em>] Report a fatal error if a character is encountered for which the
     * processor cannot determine the normalization properties. </dd>
     * </dl></dd>
     * <dt>
     * <code>"normalize-characters"</code></dt>
     * <dd> This parameter is equivalent to
     * the one defined by <code>DOMConfiguration</code> in [<a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407'>DOM Level 3 Core</a>]
     * . Unlike in the Core, the default value for this parameter is
     * <code>true</code>. While DOM implementations are not required to
     * support <a href='http://www.w3.org/TR/2004/REC-xml11-20040204/#dt-fullnorm'>fully
     * normalizing</a> the characters in the document according to appendix E of [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>], this
     * parameter must be activated by default if supported. </dd>
     * <dt>
     * <code>"xml-declaration"</code></dt>
     * <dd>
     * <dl>
     * <dt><code>true</code></dt>
```

```
     * <dd>[<em>required</em>] (<em>default</em>) If a <code>Document</code>, <code>Element</code>,
         or <code>Entity</code>
     *  node is serialized, the XML declaration, or text declaration, should
     * be included. The version (<code>Document.xmlVersion</code> if the
     * document is a Level 3 document and the version is non-null, otherwise
     * use the value "1.0"), and the output encoding (see
     * <code>LSSerializer.write</code> for details on how to find the output
     * encoding) are specified in the serialized XML declaration. </dd>
     * <dt>
     * <code>false</code></dt>
     * <dd>[<em>required</em>] Do not serialize the XML and text declarations. Report a
     * <code>"xml-declaration-needed"</code> warning if this will cause
     * problems (i.e. the serialized data is of an XML version other than [<a href='http://www.w3.
         org/TR/2004/REC-xml-20040204'>XML 1.0</a>], or an
     * encoding would be needed to be able to re-parse the serialized data). </dd>
     * </dl></dd>
     * </dl>
     */
    public DOMConfiguration getDomConfig();

    /**
     *  The end-of-line sequence of characters to be used in the XML being
     * written out. Any string is supported, but XML treats only a certain
     * set of characters sequence as end-of-line (See section 2.11,
     * "End-of-Line Handling" in [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a
         >], if the
     * serialized content is XML 1.0 or section 2.11, "End-of-Line Handling"
     * in [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>], if the
     * serialized content is XML 1.1). Using other character sequences than
     * the recommended ones can result in a document that is either not
     * serializable or not well-formed).
     * <br> On retrieval, the default value of this attribute is the
     * implementation specific default end-of-line sequence. DOM
     * implementations should choose the default to match the usual
     * convention for text files in the environment being used.
     * Implementations must choose a default sequence that matches one of
     * those allowed by XML 1.0 or XML 1.1, depending on the serialized
     * content. Setting this attribute to <code>null</code> will reset its
     * value to the default value.
     * <br>
     */
    public String getNewLine();
    /**
     *  The end-of-line sequence of characters to be used in the XML being
     * written out. Any string is supported, but XML treats only a certain
     * set of characters sequence as end-of-line (See section 2.11,
     * "End-of-Line Handling" in [<a href='http://www.w3.org/TR/2004/REC-xml-20040204'>XML 1.0</a
         >], if the
     * serialized content is XML 1.0 or section 2.11, "End-of-Line Handling"
     * in [<a href='http://www.w3.org/TR/2004/REC-xml11-20040204/'>XML 1.1</a>], if the
     * serialized content is XML 1.1). Using other character sequences than
     * the recommended ones can result in a document that is either not
     * serializable or not well-formed).
```

```
     * <br> On retrieval, the default value of this attribute is the
     * implementation specific default end-of-line sequence. DOM
     * implementations should choose the default to match the usual
     * convention for text files in the environment being used.
     * Implementations must choose a default sequence that matches one of
     * those allowed by XML 1.0 or XML 1.1, depending on the serialized
     * content. Setting this attribute to <code>null</code> will reset its
     * value to the default value.
     * <br>
     */
    public void setNewLine(String newLine);

    /**
     *  When the application provides a filter, the serializer will call out
     * to the filter before serializing each Node. The filter implementation
     * can choose to remove the node from the stream or to terminate the
     * serialization early.
     * <br> The filter is invoked after the operations requested by the
     * <code>DOMConfiguration</code> parameters have been applied. For
     * example, CDATA sections won't be passed to the filter if "<a href='http://www.w3.org/TR/DOM-
         Level-3-Core/core.html#parameter-cdata-sections'>
     * cdata-sections</a>" is set to <code>false</code>.
     */
    public LSSerializerFilter getFilter();
    /**
     *  When the application provides a filter, the serializer will call out
     * to the filter before serializing each Node. The filter implementation
     * can choose to remove the node from the stream or to terminate the
     * serialization early.
     * <br> The filter is invoked after the operations requested by the
     * <code>DOMConfiguration</code> parameters have been applied. For
     * example, CDATA sections won't be passed to the filter if "<a href='http://www.w3.org/TR/DOM-
         Level-3-Core/core.html#parameter-cdata-sections'>
     * cdata-sections</a>" is set to <code>false</code>.
     */
    public void setFilter(LSSerializerFilter filter);

    /**
     *  Serialize the specified node as described above in the general
     * description of the <code>LSSerializer</code> interface. The output is
     * written to the supplied <code>LSOutput</code>.
     * <br> When writing to a <code>LSOutput</code>, the encoding is found by
     * looking at the encoding information that is reachable through the
     * <code>LSOutput</code> and the item to be written (or its owner
     * document) in this order:
     * <ol>
     * <li> <code>LSOutput.encoding</code>,
     * </li>
     * <li>
     * <code>Document.inputEncoding</code>,
     * </li>
     * <li>
     * <code>Document.xmlEncoding</code>.
```

```
     * </li>
     * </ol>
     * <br> If no encoding is reachable through the above properties, a
     * default encoding of "UTF-8" will be used. If the specified encoding
     * is not supported an "unsupported-encoding" fatal error is raised.
     * <br> If no output is specified in the <code>LSOutput</code>, a
     * "no-output-specified" fatal error is raised.
     * <br> The implementation is responsible of associating the appropriate
     * media type with the serialized data.
     * <br> When writing to a HTTP URI, a HTTP PUT is performed. When writing
     * to other types of URIs, the mechanism for writing the data to the URI
     * is implementation dependent.
     * @param nodeArg  The node to serialize.
     * @param destination The destination for the serialized DOM.
     * @return  Returns <code>true</code> if <code>node</code> was
     *   successfully serialized. Return <code>false</code> in case the
     *   normal processing stopped but the implementation kept serializing
     *   the document; the result of the serialization being implementation
     *   dependent then.
     * @exception LSException
     *    SERIALIZE_ERR: Raised if the <code>LSSerializer</code> was unable to
     *   serialize the node. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level
         -3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
    public boolean write(Node nodeArg,
                         LSOutput destination)
                         throws LSException;

    /**
     *  A convenience method that acts as if <code>LSSerializer.write</code>
     * was called with a <code>LSOutput</code> with no encoding specified
     * and <code>LSOutput.systemId</code> set to the <code>uri</code>
     * argument.
     * @param nodeArg  The node to serialize.
     * @param uri The URI to write to.
     * @return  Returns <code>true</code> if <code>node</code> was
     *   successfully serialized. Return <code>false</code> in case the
     *   normal processing stopped but the implementation kept serializing
     *   the document; the result of the serialization being implementation
     *   dependent then.
     * @exception LSException
     *    SERIALIZE_ERR: Raised if the <code>LSSerializer</code> was unable to
     *   serialize the node. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level
         -3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
    public boolean writeToURI(Node nodeArg,
                              String uri)
                              throws LSException;

```

```
    /**
     *  Serialize the specified node as described above in the general
     * description of the <code>LSSerializer</code> interface. The output is
     * written to a <code>DOMString</code> that is returned to the caller.
     * The encoding used is the encoding of the <code>DOMString</code> type,
     * i.e. UTF-16. Note that no Byte Order Mark is generated in a
     * <code>DOMString</code> object.
     * @param nodeArg  The node to serialize.
     * @return  Returns the serialized data.
     * @exception DOMException
     *    DOMSTRING_SIZE_ERR: Raised if the resulting string is too long to
     *   fit in a <code>DOMString</code>.
     * @exception LSException
     *    SERIALIZE_ERR: Raised if the <code>LSSerializer</code> was unable to
     *   serialize the node. DOM applications should attach a
     *   <code>DOMErrorHandler</code> using the parameter "<a href='http://www.w3.org/TR/DOM-Level
         -3-Core/core.html#parameter-error-handler'>
     *   error-handler</a>" if they wish to get details on the error.
     */
    public String writeToString(Node nodeArg)
                                throws DOMException, LSException;

}
```

41.11 LSSerializerFilter.java

Secuencia 457: org/w3c/dom/ls/LSSerializerFilter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
```

```
 *
 *
 *
 *
 * Copyright (c) 2004 World Wide Web Consortium,
 *
 * (Massachusetts Institute of Technology, European Research Consortium for
 * Informatics and Mathematics, Keio University). All Rights Reserved. This
 * work is distributed under the W3C(r) Software License [1] in the hope that
 * it will be useful, but WITHOUT ANY WARRANTY; without even the implied
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * [1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
 */

package org.w3c.dom.ls;

import org.w3c.dom.traversal.NodeFilter;

/**
 *  <code>LSSerializerFilter</code>s provide applications the ability to
 * examine nodes as they are being serialized and decide what nodes should
 * be serialized or not. The <code>LSSerializerFilter</code> interface is
 * based on the <code>NodeFilter</code> interface defined in [<a href='http://www.w3.org/TR/2000/
     REC-DOM-Level-2-Traversal-Range-20001113'>DOM Level 2 Traversal and      Range</a>]
 * .
 * <p> <code>Document</code>, <code>DocumentType</code>,
 * <code>DocumentFragment</code>, <code>Notation</code>, <code>Entity</code>
 * , and children of <code>Attr</code> nodes are not passed to the filter.
 * The child nodes of an <code>EntityReference</code> node are only passed
 * to the filter if the <code>EntityReference</code> node is skipped by the
 * method <code>LSParserFilter.acceptNode()</code>.
 * <p> When serializing an <code>Element</code>, the element is passed to the
 * filter before any of its attributes are passed to the filter. Namespace
 * declaration attributes, and default attributes (except in the case when "
 * discard-default-content" is set to <code>false</code>), are never passed
 * to the filter.
 * <p> The result of any attempt to modify a node passed to a
 * <code>LSSerializerFilter</code> is implementation dependent.
 * <p> DOM applications must not raise exceptions in a filter. The effect of
 * throwing exceptions from a filter is DOM implementation dependent.
 * <p> For efficiency, a node passed to the filter may not be the same as the
 * one that is actually in the tree. And the actual node (node object
 * identity) may be reused during the process of filtering and serializing a
 * document.
 * <p>See also the <a href='http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407'>Document Object
     Model (DOM) Level 3 Load
and Save Specification</a>.
 */
public interface LSSerializerFilter extends NodeFilter {
    /**
     *  Tells the <code>LSSerializer</code> what types of nodes to show to the
     * filter. If a node is not shown to the filter using this attribute, it
```

```
     * is automatically serialized. See <code>NodeFilter</code> for
     * definition of the constants. The constants <code>SHOW_DOCUMENT</code>
     * , <code>SHOW_DOCUMENT_TYPE</code>, <code>SHOW_DOCUMENT_FRAGMENT</code>
     * , <code>SHOW_NOTATION</code>, and <code>SHOW_ENTITY</code> are
     * meaningless here, such nodes will never be passed to a
     * <code>LSSerializerFilter</code>.
     * <br> Unlike [<a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
         DOM Level 2 Traversal and      Range</a>]
     * , the <code>SHOW_ATTRIBUTE</code> constant indicates that the
     * <code>Attr</code> nodes are shown and passed to the filter.
     * <br> The constants used here are defined in [<a href='http://www.w3.org/TR/2000/REC-DOM-
         Level-2-Traversal-Range-20001113'>DOM Level 2 Traversal and      Range</a>]
     * .
     */
    public int getWhatToShow();

}
```

42 Dom Ranges (org.w3c.dom.ranges package)

42.1 DocumentRange.java

Secuencia 458: org/w3c/dom/ranges/DocumentRange.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
```

```
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.ranges;

/**
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public interface DocumentRange {
    /**
     * This interface can be obtained from the object implementing the
     * <code>Document</code> interface using binding-specific casting
     * methods.
     * @return The initial state of the Range returned from this method is
     *   such that both of its boundary-points are positioned at the
     *   beginning of the corresponding Document, before any content. The
     *   Range returned can only be used to select content associated with
     *   this Document, or with DocumentFragments and Attrs for which this
     *   Document is the <code>ownerDocument</code>.
     */
    public Range createRange();

}
```

42.2 Range.java

Secuencia 459: org/w3c/dom/ranges/Range.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.ranges;

import org.w3c.dom.Node;
import org.w3c.dom.DOMException;
import org.w3c.dom.DocumentFragment;

/**
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public interface Range {
    /**
     * Node within which the Range begins
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public Node getStartContainer()
                        throws DOMException;

    /**
     * Offset within the starting node of the Range.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public int getStartOffset()
```

```
                    throws DOMException;

    /**
     * Node within which the Range ends
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public Node getEndContainer()
                    throws DOMException;

    /**
     * Offset within the ending node of the Range.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public int getEndOffset()
                    throws DOMException;

    /**
     * TRUE if the Range is collapsed
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public boolean getCollapsed()
                    throws DOMException;

    /**
     * The deepest common ancestor container of the Range's two
     * boundary-points.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public Node getCommonAncestorContainer()
                    throws DOMException;

    /**
     * Sets the attributes describing the start of the Range.
     * @param refNode The <code>refNode</code> value. This parameter must be
     *   different from <code>null</code>.
     * @param offset The <code>startOffset</code> value.
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if <code>refNode</code> or an ancestor
     *   of <code>refNode</code> is an Entity, Notation, or DocumentType
     *   node.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if <code>offset</code> is negative or greater
     *   than the number of child units in <code>refNode</code>. Child units
     *   are 16-bit units if <code>refNode</code> is a type of CharacterData
     *   node (e.g., a Text or Comment node) or a ProcessingInstruction
```

```
     *   node. Child units are Nodes in all other cases.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setStart(Node refNode,
                         int offset)
                         throws RangeException, DOMException;

    /**
     * Sets the attributes describing the end of a Range.
     * @param refNode The <code>refNode</code> value. This parameter must be
     *   different from <code>null</code>.
     * @param offset The <code>endOffset</code> value.
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if <code>refNode</code> or an ancestor
     *   of <code>refNode</code> is an Entity, Notation, or DocumentType
     *   node.
     * @exception DOMException
     *   INDEX_SIZE_ERR: Raised if <code>offset</code> is negative or greater
     *   than the number of child units in <code>refNode</code>. Child units
     *   are 16-bit units if <code>refNode</code> is a type of CharacterData
     *   node (e.g., a Text or Comment node) or a ProcessingInstruction
     *   node. Child units are Nodes in all other cases.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setEnd(Node refNode,
                       int offset)
                       throws RangeException, DOMException;

    /**
     * Sets the start position to be before a node
     * @param refNode Range starts before <code>refNode</code>
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if the root container of
     *   <code>refNode</code> is not an Attr, Document, or DocumentFragment
     *   node or if <code>refNode</code> is a Document, DocumentFragment,
     *   Attr, Entity, or Notation node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setStartBefore(Node refNode)
                               throws RangeException, DOMException;

    /**
     * Sets the start position to be after a node
```

```
     * @param refNode Range starts after <code>refNode</code>
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if the root container of
     *   <code>refNode</code> is not an Attr, Document, or DocumentFragment
     *   node or if <code>refNode</code> is a Document, DocumentFragment,
     *   Attr, Entity, or Notation node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setStartAfter(Node refNode)
                            throws RangeException, DOMException;

    /**
     * Sets the end position to be before a node.
     * @param refNode Range ends before <code>refNode</code>
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if the root container of
     *   <code>refNode</code> is not an Attr, Document, or DocumentFragment
     *   node or if <code>refNode</code> is a Document, DocumentFragment,
     *   Attr, Entity, or Notation node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setEndBefore(Node refNode)
                           throws RangeException, DOMException;

    /**
     * Sets the end of a Range to be after a node
     * @param refNode Range ends after <code>refNode</code>.
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if the root container of
     *   <code>refNode</code> is not an Attr, Document or DocumentFragment
     *   node or if <code>refNode</code> is a Document, DocumentFragment,
     *   Attr, Entity, or Notation node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void setEndAfter(Node refNode)
                          throws RangeException, DOMException;

    /**
     * Collapse a Range onto one of its boundary-points
     * @param toStart If TRUE, collapses the Range onto its start; if FALSE,
     *   collapses it onto its end.
```

```
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public void collapse(boolean toStart)
                         throws DOMException;

    /**
     * Select a node and its contents
     * @param refNode The node to select.
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if an ancestor of <code>refNode</code>
     *   is an Entity, Notation or DocumentType node or if
     *   <code>refNode</code> is a Document, DocumentFragment, Attr, Entity,
     *   or Notation node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void selectNode(Node refNode)
                           throws RangeException, DOMException;

    /**
     * Select the contents within a node
     * @param refNode Node to select from
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if <code>refNode</code> or an ancestor
     *   of <code>refNode</code> is an Entity, Notation or DocumentType node.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>refNode</code> was created
     *   from a different document than the one that created this range.
     */
    public void selectNodeContents(Node refNode)
                                   throws RangeException, DOMException;

    // CompareHow
    /**
     * Compare start boundary-point of <code>sourceRange</code> to start
     * boundary-point of Range on which <code>compareBoundaryPoints</code>
     * is invoked.
     */
    public static final short START_TO_START            = 0;
    /**
     * Compare start boundary-point of <code>sourceRange</code> to end
     * boundary-point of Range on which <code>compareBoundaryPoints</code>
     * is invoked.
     */
    public static final short START_TO_END              = 1;
    /**
```

```
     * Compare end boundary-point of <code>sourceRange</code> to end
     * boundary-point of Range on which <code>compareBoundaryPoints</code>
     * is invoked.
     */
    public static final short END_TO_END                = 2;
    /**
     * Compare end boundary-point of <code>sourceRange</code> to start
     * boundary-point of Range on which <code>compareBoundaryPoints</code>
     * is invoked.
     */
    public static final short END_TO_START              = 3;

    /**
     * Compare the boundary-points of two Ranges in a document.
     * @param how A code representing the type of comparison, as defined
     *   above.
     * @param sourceRange The <code>Range</code> on which this current
     *   <code>Range</code> is compared to.
     * @return  -1, 0 or 1 depending on whether the corresponding
     *   boundary-point of the Range is respectively before, equal to, or
     *   after the corresponding boundary-point of <code>sourceRange</code>.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: Raised if the two Ranges are not in the same
     *   Document or DocumentFragment.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     */
    public short compareBoundaryPoints(short how,
                                       Range sourceRange)
                                       throws DOMException;

    /**
     * Removes the contents of a Range from the containing document or
     * document fragment without returning a reference to the removed
     * content.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if any portion of the content of
     *   the Range is read-only or any of the nodes that contain any of the
     *   content of the Range are read-only.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     */
    public void deleteContents()
                              throws DOMException;

    /**
     * Moves the contents of a Range from the containing document or document
     * fragment to a new DocumentFragment.
     * @return A DocumentFragment containing the extracted contents.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if any portion of the content of
     *   the Range is read-only or any of the nodes which contain any of the
     *   content of the Range are read-only.
```

```
     *   <br>HIERARCHY_REQUEST_ERR: Raised if a DocumentType node would be
     *   extracted into the new DocumentFragment.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     */
    public DocumentFragment extractContents()
                                          throws DOMException;

    /**
     * Duplicates the contents of a Range
     * @return A DocumentFragment that contains content equivalent to this
     *   Range.
     * @exception DOMException
     *   HIERARCHY_REQUEST_ERR: Raised if a DocumentType node would be
     *   extracted into the new DocumentFragment.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     */
    public DocumentFragment cloneContents()
                                        throws DOMException;

    /**
     * Inserts a node into the Document or DocumentFragment at the start of
     * the Range. If the container is a Text node, this will be split at the
     * start of the Range (as if the Text node's splitText method was
     * performed at the insertion point) and the insertion will occur
     * between the two resulting Text nodes. Adjacent Text nodes will not be
     * automatically merged. If the node to be inserted is a
     * DocumentFragment node, the children will be inserted rather than the
     * DocumentFragment node itself.
     * @param newNode The node to insert at the start of the Range
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if an ancestor container of the
     *   start of the Range is read-only.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code>newNode</code> and the
     *   container of the start of the Range were not created from the same
     *   document.
     *   <br>HIERARCHY_REQUEST_ERR: Raised if the container of the start of
     *   the Range is of a type that does not allow children of the type of
     *   <code>newNode</code> or if <code>newNode</code> is an ancestor of
     *   the container.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     * @exception RangeException
     *   INVALID_NODE_TYPE_ERR: Raised if <code>newNode</code> is an Attr,
     *   Entity, Notation, or Document node.
     */
    public void insertNode(Node newNode)
                           throws DOMException, RangeException;

    /**
     * Reparents the contents of the Range to the given node and inserts the
     * node at the position of the start of the Range.
```

```
     * @param newParent The node to surround the contents with.
     * @exception DOMException
     *   NO_MODIFICATION_ALLOWED_ERR: Raised if an ancestor container of
     *   either boundary-point of the Range is read-only.
     *   <br>WRONG_DOCUMENT_ERR: Raised if <code> newParent</code> and the
     *   container of the start of the Range were not created from the same
     *   document.
     *   <br>HIERARCHY_REQUEST_ERR: Raised if the container of the start of
     *   the Range is of a type that does not allow children of the type of
     *   <code>newParent</code> or if <code>newParent</code> is an ancestor
     *   of the container or if <code>node</code> would end up with a child
     *   node of a type not allowed by the type of <code>node</code>.
     *   <br>INVALID_STATE_ERR: Raised if <code>detach()</code> has already
     *   been invoked on this object.
     * @exception RangeException
     *   BAD_BOUNDARYPOINTS_ERR: Raised if the Range partially selects a
     *   non-text node.
     *   <br>INVALID_NODE_TYPE_ERR: Raised if <code> node</code> is an Attr,
     *   Entity, DocumentType, Notation, Document, or DocumentFragment node.
     */
    public void surroundContents(Node newParent)
                                throws DOMException, RangeException;

    /**
     * Produces a new Range whose boundary-points are equal to the
     * boundary-points of the Range.
     * @return The duplicated Range.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public Range cloneRange()
                           throws DOMException;

    /**
     * Returns the contents of a Range as a string. This string contains only
     * the data characters, not any markup.
     * @return The contents of the Range.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
     *   invoked on this object.
     */
    public String toString()
                          throws DOMException;

    /**
     * Called to indicate that the Range is no longer in use and that the
     * implementation may relinquish any resources associated with this
     * Range. Subsequent calls to any methods or attribute getters on this
     * Range will result in a <code>DOMException</code> being thrown with an
     * error code of <code>INVALID_STATE_ERR</code>.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if <code>detach()</code> has already been
```

```
     *   invoked on this object.
     */
    public void detach()
                        throws DOMException;

}
```

42.3 RangeException.java

Secuencia 460: org/w3c/dom/ranges/RangeException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.ranges;

```

```
/**
 * Range operations may throw a <code>RangeException</code> as specified in
 * their method descriptions.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public class RangeException extends RuntimeException {
    public RangeException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
    // RangeExceptionCode
    /**
     * If the boundary-points of a Range do not meet specific requirements.
     */
    public static final short BAD_BOUNDARYPOINTS_ERR    = 1;
    /**
     * If the container of an boundary-point of a Range is being set to either
     * a node of an invalid type or a node with an ancestor of an invalid
     * type.
     */
    public static final short INVALID_NODE_TYPE_ERR     = 2;

}
```

43 Dom Stylesheets (org.w3c.dom.stylesheets package)

43.1 DocumentStyle.java

Secuencia 461: org/w3c/dom/stylesheets/DocumentStyle.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.stylesheets;

/**
 *  The <code>DocumentStyle</code> interface provides a mechanism by which the
 * style sheets embedded in a document can be retrieved. The expectation is
 * that an instance of the <code>DocumentStyle</code> interface can be
 * obtained by using binding-specific casting methods on an instance of the
 * <code>Document</code> interface.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface DocumentStyle {
    /**
     *  A list containing all the style sheets explicitly linked into or
     * embedded in a document. For HTML documents, this includes external
     * style sheets, included via the HTML  LINK element, and inline  STYLE
     * elements. In XML, this includes external style sheets, included via
     * style sheet processing instructions (see [XML StyleSheet]).
     */
    public StyleSheetList getStyleSheets();

}
```

43.2 LinkStyle.java

Secuencia 462: org/w3c/dom/stylesheets/LinkStyle.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.stylesheets;

/**
 *  The <code>LinkStyle</code> interface provides a mechanism by which a style
 * sheet can be retrieved from the node responsible for linking it into a
 * document. An instance of the <code>LinkStyle</code> interface can be
 * obtained using binding-specific casting methods on an instance of a
 * linking node (<code>HTMLLinkElement</code>, <code>HTMLStyleElement</code>
 * or <code>ProcessingInstruction</code> in DOM Level 2).
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface LinkStyle {
    /**
     *  The style sheet.
     */
    public StyleSheet getSheet();
```

```

}
```

43.3 MediaList.java

Secuencia 463: org/w3c/dom/stylesheets/MediaList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.stylesheets;

import org.w3c.dom.DOMException;

/**
 *  The <code>MediaList</code> interface provides the abstraction of an
```

```
 * ordered collection of media, without defining or constraining how this
 * collection is implemented. An empty list is the same as a list that
 * contains the medium <code>"all"</code>.
 * <p> The items in the <code>MediaList</code> are accessible via an integral
 * index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface MediaList {
    /**
     *  The parsable textual representation of the media list. This is a
     * comma-separated list of media.
     */
    public String getMediaText();
    /**
     *  The parsable textual representation of the media list. This is a
     * comma-separated list of media.
     * @exception DOMException
     *   SYNTAX_ERR: Raised if the specified string value has a syntax error
     *   and is unparsable.
     *   <br>NO_MODIFICATION_ALLOWED_ERR: Raised if this media list is
     *   readonly.
     */
    public void setMediaText(String mediaText)
                            throws DOMException;

    /**
     *  The number of media in the list. The range of valid media is
     * <code>0</code> to <code>length-1</code> inclusive.
     */
    public int getLength();

    /**
     *  Returns the <code>index</code>th in the list. If <code>index</code> is
     * greater than or equal to the number of media in the list, this
     * returns <code>null</code>.
     * @param index  Index into the collection.
     * @return  The medium at the <code>index</code>th position in the
     *   <code>MediaList</code>, or <code>null</code> if that is not a valid
     *   index.
     */
    public String item(int index);

    /**
     *  Deletes the medium indicated by <code>oldMedium</code> from the list.
     * @param oldMedium The medium to delete in the media list.
     * @exception DOMException
     *    NO_MODIFICATION_ALLOWED_ERR: Raised if this list is readonly.
     *   <br> NOT_FOUND_ERR: Raised if <code>oldMedium</code> is not in the
     *   list.
     */
    public void deleteMedium(String oldMedium)
```

```
                            throws DOMException;

    /**
     *  Adds the medium <code>newMedium</code> to the end of the list. If the
     * <code>newMedium</code> is already used, it is first removed.
     * @param newMedium The new medium to add.
     * @exception DOMException
     *    INVALID_CHARACTER_ERR: If the medium contains characters that are
     *   invalid in the underlying style language.
     *   <br> NO_MODIFICATION_ALLOWED_ERR: Raised if this list is readonly.
     */
    public void appendMedium(String newMedium)
                            throws DOMException;

}
```

43.4 StyleSheet.java

Secuencia 464: org/w3c/dom/stylesheets/StyleSheet.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
```

```
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.stylesheets;

import org.w3c.dom.Node;

/**
 *  The <code>StyleSheet</code> interface is the abstract base interface for
 * any type of style sheet. It represents a single style sheet associated
 * with a structured document. In HTML, the StyleSheet interface represents
 * either an external style sheet, included via the HTML  LINK element, or
 * an inline  STYLE element. In XML, this interface represents an external
 * style sheet, included via a style sheet processing instruction.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface StyleSheet {
    /**
     *  This specifies the style sheet language for this style sheet. The
     * style sheet language is specified as a content type (e.g.
     * "text/css"). The content type is often specified in the
     * <code>ownerNode</code>. Also see the type attribute definition for
     * the <code>LINK</code> element in HTML 4.0, and the type
     * pseudo-attribute for the XML style sheet processing instruction.
     */
    public String getType();

    /**
     *  <code>false</code> if the style sheet is applied to the document.
     * <code>true</code> if it is not. Modifying this attribute may cause a
     * new resolution of style for the document. A stylesheet only applies
     * if both an appropriate medium definition is present and the disabled
     * attribute is false. So, if the media doesn't apply to the current
     * user agent, the <code>disabled</code> attribute is ignored.
     */
    public boolean getDisabled();
    /**
     *  <code>false</code> if the style sheet is applied to the document.
     * <code>true</code> if it is not. Modifying this attribute may cause a
     * new resolution of style for the document. A stylesheet only applies
     * if both an appropriate medium definition is present and the disabled
     * attribute is false. So, if the media doesn't apply to the current
     * user agent, the <code>disabled</code> attribute is ignored.
     */
    public void setDisabled(boolean disabled);

    /**
```

```
     *  The node that associates this style sheet with the document. For HTML,
     * this may be the corresponding <code>LINK</code> or <code>STYLE</code>
     * element. For XML, it may be the linking processing instruction. For
     * style sheets that are included by other style sheets, the value of
     * this attribute is <code>null</code>.
     */
    public Node getOwnerNode();

    /**
     *  For style sheet languages that support the concept of style sheet
     * inclusion, this attribute represents the including style sheet, if
     * one exists. If the style sheet is a top-level style sheet, or the
     * style sheet language does not support inclusion, the value of this
     * attribute is <code>null</code>.
     */
    public StyleSheet getParentStyleSheet();

    /**
     *  If the style sheet is a linked style sheet, the value of its attribute
     * is its location. For inline style sheets, the value of this attribute
     * is <code>null</code>. See the href attribute definition for the
     * <code>LINK</code> element in HTML 4.0, and the href pseudo-attribute
     * for the XML style sheet processing instruction.
     */
    public String getHref();

    /**
     *  The advisory title. The title is often specified in the
     * <code>ownerNode</code>. See the title attribute definition for the
     * <code>LINK</code> element in HTML 4.0, and the title pseudo-attribute
     * for the XML style sheet processing instruction.
     */
    public String getTitle();

    /**
     *  The intended destination media for style information. The media is
     * often specified in the <code>ownerNode</code>. If no media has been
     * specified, the <code>MediaList</code> will be empty. See the media
     * attribute definition for the <code>LINK</code> element in HTML 4.0,
     * and the media pseudo-attribute for the XML style sheet processing
     * instruction . Modifying the media list may cause a change to the
     * attribute <code>disabled</code>.
     */
    public MediaList getMedia();

}
```

43.5 StyleSheetList.java

Secuencia 465: org/w3c/dom/stylesheets/StyleSheetList.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.stylesheets;

/**
 * The <code>StyleSheetList</code> interface provides the abstraction of an
 * ordered collection of style sheets.
 * <p> The items in the <code>StyleSheetList</code> are accessible via an
 * integral index, starting from 0.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113'>Document
     Object Model (DOM) Level 2 Style Specification</a>.
 * @since DOM Level 2
 */
public interface StyleSheetList {
    /**
     *  The number of <code>StyleSheets</code> in the list. The range of valid
     * child stylesheet indices is <code>0</code> to <code>length-1</code>
```

```
     * inclusive.
     */
    public int getLength();

    /**
     *  Used to retrieve a style sheet by ordinal index. If index is greater
     * than or equal to the number of style sheets in the list, this returns
     * <code>null</code>.
     * @param index Index into the collection
     * @return The style sheet at the <code>index</code> position in the
     *   <code>StyleSheetList</code>, or <code>null</code> if that is not a
     *   valid index.
     */
    public StyleSheet item(int index);

}
```

44 Dom Traversal (org.w3c.dom.traversal package)

44.1 DocumentTraversal.java

Secuencia 466: org/w3c/dom/traversal/DocumentTraversal.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
```

```
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.traversal;

import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 * <code>DocumentTraversal</code> contains methods that create
 * <code>NodeIterators</code> and <code>TreeWalkers</code> to traverse a
 * node and its children in document order (depth first, pre-order
 * traversal, which is equivalent to the order in which the start tags occur
 * in the text representation of the document). In DOMs which support the
 * Traversal feature, <code>DocumentTraversal</code> will be implemented by
 * the same objects that implement the Document interface.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public interface DocumentTraversal {
    /**
     * Create a new <code>NodeIterator</code> over the subtree rooted at the
     * specified node.
     * @param root The node which will be iterated together with its
     *   children. The <code>NodeIterator</code> is initially positioned
     *   just before this node. The <code>whatToShow</code> flags and the
     *   filter, if any, are not considered when setting this position. The
     *   root must not be <code>null</code>.
     * @param whatToShow This flag specifies which node types may appear in
     *   the logical view of the tree presented by the
     *   <code>NodeIterator</code>. See the description of
     *   <code>NodeFilter</code> for the set of possible <code>SHOW_</code>
     *   values.These flags can be combined using <code>OR</code>.
     * @param filter The <code>NodeFilter</code> to be used with this
     *   <code>NodeIterator</code>, or <code>null</code> to indicate no
     *   filter.
     * @param entityReferenceExpansion The value of this flag determines
     *   whether entity reference nodes are expanded.
     * @return The newly created <code>NodeIterator</code>.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if the specified <code>root</code> is
     *   <code>null</code>.
     */
    public NodeIterator createNodeIterator(Node root,
                                           int whatToShow,
```

```
                                        NodeFilter filter,
                                        boolean entityReferenceExpansion)
                                        throws DOMException;

    /**
     * Create a new <code>TreeWalker</code> over the subtree rooted at the
     * specified node.
     * @param root The node which will serve as the <code>root</code> for the
     *   <code>TreeWalker</code>. The <code>whatToShow</code> flags and the
     *   <code>NodeFilter</code> are not considered when setting this value;
     *   any node type will be accepted as the <code>root</code>. The
     *   <code>currentNode</code> of the <code>TreeWalker</code> is
     *   initialized to this node, whether or not it is visible. The
     *   <code>root</code> functions as a stopping point for traversal
     *   methods that look upward in the document structure, such as
     *   <code>parentNode</code> and nextNode. The <code>root</code> must
     *   not be <code>null</code>.
     * @param whatToShow This flag specifies which node types may appear in
     *   the logical view of the tree presented by the
     *   <code>TreeWalker</code>. See the description of
     *   <code>NodeFilter</code> for the set of possible <code>SHOW_</code>
     *   values.These flags can be combined using <code>OR</code>.
     * @param filter The <code>NodeFilter</code> to be used with this
     *   <code>TreeWalker</code>, or <code>null</code> to indicate no filter.
     * @param entityReferenceExpansion If this flag is false, the contents of
     *   <code>EntityReference</code> nodes are not presented in the logical
     *   view.
     * @return The newly created <code>TreeWalker</code>.
     * @exception DOMException
     *    NOT_SUPPORTED_ERR: Raised if the specified <code>root</code> is
     *   <code>null</code>.
     */
    public TreeWalker createTreeWalker(Node root,
                                       int whatToShow,
                                       NodeFilter filter,
                                       boolean entityReferenceExpansion)
                                       throws DOMException;

}
```

44.2 NodeFilter.java

Secuencia 467: org/w3c/dom/traversal/NodeFilter.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.traversal;

import org.w3c.dom.Node;

/**
 * Filters are objects that know how to "filter out" nodes. If a
 * <code>NodeIterator</code> or <code>TreeWalker</code> is given a
 * <code>NodeFilter</code>, it applies the filter before it returns the next
 * node. If the filter says to accept the node, the traversal logic returns
 * it; otherwise, traversal looks for the next node and pretends that the
 * node that was rejected was not there.
 * <p>The DOM does not provide any filters. <code>NodeFilter</code> is just an
 * interface that users can implement to provide their own filters.
 * <p><code>NodeFilters</code> do not need to know how to traverse from node
 * to node, nor do they need to know anything about the data structure that
 * is being traversed. This makes it very easy to write filters, since the
 * only thing they have to know how to do is evaluate a single node. One
 * filter may be used with a number of different kinds of traversals,
 * encouraging code reuse.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
```

```
 */
public interface NodeFilter {
    // Constants returned by acceptNode
    /**
     * Accept the node. Navigation methods defined for
     * <code>NodeIterator</code> or <code>TreeWalker</code> will return this
     * node.
     */
    public static final short FILTER_ACCEPT             = 1;
    /**
     * Reject the node. Navigation methods defined for
     * <code>NodeIterator</code> or <code>TreeWalker</code> will not return
     * this node. For <code>TreeWalker</code>, the children of this node
     * will also be rejected. <code>NodeIterators</code> treat this as a
     * synonym for <code>FILTER_SKIP</code>.
     */
    public static final short FILTER_REJECT             = 2;
    /**
     * Skip this single node. Navigation methods defined for
     * <code>NodeIterator</code> or <code>TreeWalker</code> will not return
     * this node. For both <code>NodeIterator</code> and
     * <code>TreeWalker</code>, the children of this node will still be
     * considered.
     */
    public static final short FILTER_SKIP               = 3;

    // Constants for whatToShow
    /**
     * Show all <code>Nodes</code>.
     */
    public static final int SHOW_ALL                  = 0xFFFFFFFF;
    /**
     * Show <code>Element</code> nodes.
     */
    public static final int SHOW_ELEMENT              = 0x00000001;
    /**
     * Show <code>Attr</code> nodes. This is meaningful only when creating an
     * <code>NodeIterator</code> or <code>TreeWalker</code> with an
     * attribute node as its <code>root</code>; in this case, it means that
     * the attribute node will appear in the first position of the iteration
     * or traversal. Since attributes are never children of other nodes,
     * they do not appear when traversing over the document tree.
     */
    public static final int SHOW_ATTRIBUTE            = 0x00000002;
    /**
     * Show <code>Text</code> nodes.
     */
    public static final int SHOW_TEXT                 = 0x00000004;
    /**
     * Show <code>CDATASection</code> nodes.
     */
    public static final int SHOW_CDATA_SECTION        = 0x00000008;
    /**
```

```
     * Show <code>EntityReference</code> nodes.
     */
    public static final int SHOW_ENTITY_REFERENCE     = 0x00000010;
    /**
     * Show <code>Entity</code> nodes. This is meaningful only when creating
     * an <code>NodeIterator</code> or <code>TreeWalker</code> with an
     * <code>Entity</code> node as its <code>root</code>; in this case, it
     * means that the <code>Entity</code> node will appear in the first
     * position of the traversal. Since entities are not part of the
     * document tree, they do not appear when traversing over the document
     * tree.
     */
    public static final int SHOW_ENTITY               = 0x00000020;
    /**
     * Show <code>ProcessingInstruction</code> nodes.
     */
    public static final int SHOW_PROCESSING_INSTRUCTION = 0x00000040;
    /**
     * Show <code>Comment</code> nodes.
     */
    public static final int SHOW_COMMENT              = 0x00000080;
    /**
     * Show <code>Document</code> nodes.
     */
    public static final int SHOW_DOCUMENT             = 0x00000100;
    /**
     * Show <code>DocumentType</code> nodes.
     */
    public static final int SHOW_DOCUMENT_TYPE        = 0x00000200;
    /**
     * Show <code>DocumentFragment</code> nodes.
     */
    public static final int SHOW_DOCUMENT_FRAGMENT    = 0x00000400;
    /**
     * Show <code>Notation</code> nodes. This is meaningful only when creating
     * an <code>NodeIterator</code> or <code>TreeWalker</code> with a
     * <code>Notation</code> node as its <code>root</code>; in this case, it
     * means that the <code>Notation</code> node will appear in the first
     * position of the traversal. Since notations are not part of the
     * document tree, they do not appear when traversing over the document
     * tree.
     */
    public static final int SHOW_NOTATION             = 0x00000800;

    /**
     * Test whether a specified node is visible in the logical view of a
     * <code>TreeWalker</code> or <code>NodeIterator</code>. This function
     * will be called by the implementation of <code>TreeWalker</code> and
     * <code>NodeIterator</code>; it is not normally called directly from
     * user code. (Though you could do so if you wanted to use the same
     * filter to guide your own application logic.)
     * @param n The node to check to see if it passes the filter or not.
     * @return A constant to determine whether the node is accepted,
```

```
     *   rejected, or skipped, as defined above.
     */
    public short acceptNode(Node n);

}
```

44.3 NodeIterator.java

Secuencia 468: org/w3c/dom/traversal/NodeIterator.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.traversal;

import org.w3c.dom.Node;
```

```java
import org.w3c.dom.DOMException;

/**
 * <code>NodeIterators</code> are used to step through a set of nodes, e.g.
 * the set of nodes in a <code>NodeList</code>, the document subtree
 * governed by a particular <code>Node</code>, the results of a query, or
 * any other set of nodes. The set of nodes to be iterated is determined by
 * the implementation of the <code>NodeIterator</code>. DOM Level 2
 * specifies a single <code>NodeIterator</code> implementation for
 * document-order traversal of a document subtree. Instances of these
 * <code>NodeIterators</code> are created by calling
 * <code>DocumentTraversal</code><code>.createNodeIterator()</code>.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public interface NodeIterator {
    /**
     * The root node of the <code>NodeIterator</code>, as specified when it
     * was created.
     */
    public Node getRoot();

    /**
     * This attribute determines which node types are presented via the
     * <code>NodeIterator</code>. The available set of constants is defined
     * in the <code>NodeFilter</code> interface.  Nodes not accepted by
     * <code>whatToShow</code> will be skipped, but their children may still
     * be considered. Note that this skip takes precedence over the filter,
     * if any.
     */
    public int getWhatToShow();

    /**
     * The <code>NodeFilter</code> used to screen nodes.
     */
    public NodeFilter getFilter();

    /**
     *  The value of this flag determines whether the children of entity
     * reference nodes are visible to the <code>NodeIterator</code>. If
     * false, these children  and their descendants will be rejected. Note
     * that this rejection takes precedence over <code>whatToShow</code> and
     * the filter. Also note that this is currently the only situation where
     * <code>NodeIterators</code> may reject a complete subtree rather than
     * skipping individual nodes.
     * <br>
     * <br> To produce a view of the document that has entity references
     * expanded and does not expose the entity reference node itself, use
     * the <code>whatToShow</code> flags to hide the entity reference node
     * and set <code>expandEntityReferences</code> to true when creating the
     * <code>NodeIterator</code>. To produce a view of the document that has
     * entity reference nodes but no entity expansion, use the
```

```
     * <code>whatToShow</code> flags to show the entity reference node and
     * set <code>expandEntityReferences</code> to false.
     */
    public boolean getExpandEntityReferences();

    /**
     * Returns the next node in the set and advances the position of the
     * <code>NodeIterator</code> in the set. After a
     * <code>NodeIterator</code> is created, the first call to
     * <code>nextNode()</code> returns the first node in the set.
     * @return The next <code>Node</code> in the set being iterated over, or
     *   <code>null</code> if there are no more members in that set.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if this method is called after the
     *   <code>detach</code> method was invoked.
     */
    public Node nextNode()
                        throws DOMException;

    /**
     * Returns the previous node in the set and moves the position of the
     * <code>NodeIterator</code> backwards in the set.
     * @return The previous <code>Node</code> in the set being iterated over,
     *   or <code>null</code> if there are no more members in that set.
     * @exception DOMException
     *   INVALID_STATE_ERR: Raised if this method is called after the
     *   <code>detach</code> method was invoked.
     */
    public Node previousNode()
                            throws DOMException;

    /**
     * Detaches the <code>NodeIterator</code> from the set which it iterated
     * over, releasing any computational resources and placing the
     * <code>NodeIterator</code> in the INVALID state. After
     * <code>detach</code> has been invoked, calls to <code>nextNode</code>
     * or <code>previousNode</code> will raise the exception
     * INVALID_STATE_ERR.
     */
    public void detach();

}
```

44.4 TreeWalker.java

Secuencia 469: org/w3c/dom/traversal/TreeWalker.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.traversal;

import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 * <code>TreeWalker</code> objects are used to navigate a document tree or
 * subtree using the view of the document defined by their
 * <code>whatToShow</code> flags and filter (if any). Any function which
 * performs navigation using a <code>TreeWalker</code> will automatically
 * support any view defined by a <code>TreeWalker</code>.
 * <p>Omitting nodes from the logical view of a subtree can result in a
 * structure that is substantially different from the same subtree in the
 * complete, unfiltered document. Nodes that are siblings in the
 * <code>TreeWalker</code> view may be children of different, widely
 * separated nodes in the original view. For instance, consider a
 * <code>NodeFilter</code> that skips all nodes except for Text nodes and
 * the root node of a document. In the logical view that results, all text
 * nodes will be siblings and appear as direct children of the root node, no
```

```
 * matter how deeply nested the structure of the original document.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113'>
     Document Object Model (DOM) Level 2 Traversal and Range Specification</a>.
 * @since DOM Level 2
 */
public interface TreeWalker {
    /**
     * The <code>root</code> node of the <code>TreeWalker</code>, as specified
     * when it was created.
     */
    public Node getRoot();

    /**
     * This attribute determines which node types are presented via the
     * <code>TreeWalker</code>. The available set of constants is defined in
     * the <code>NodeFilter</code> interface.  Nodes not accepted by
     * <code>whatToShow</code> will be skipped, but their children may still
     * be considered. Note that this skip takes precedence over the filter,
     * if any.
     */
    public int getWhatToShow();

    /**
     * The filter used to screen nodes.
     */
    public NodeFilter getFilter();

    /**
     * The value of this flag determines whether the children of entity
     * reference nodes are visible to the <code>TreeWalker</code>. If false,
     * these children  and their descendants will be rejected. Note that
     * this rejection takes precedence over <code>whatToShow</code> and the
     * filter, if any.
     * <br> To produce a view of the document that has entity references
     * expanded and does not expose the entity reference node itself, use
     * the <code>whatToShow</code> flags to hide the entity reference node
     * and set <code>expandEntityReferences</code> to true when creating the
     * <code>TreeWalker</code>. To produce a view of the document that has
     * entity reference nodes but no entity expansion, use the
     * <code>whatToShow</code> flags to show the entity reference node and
     * set <code>expandEntityReferences</code> to false.
     */
    public boolean getExpandEntityReferences();

    /**
     * The node at which the <code>TreeWalker</code> is currently positioned.
     * <br>Alterations to the DOM tree may cause the current node to no longer
     * be accepted by the <code>TreeWalker</code>'s associated filter.
     * <code>currentNode</code> may also be explicitly set to any node,
     * whether or not it is within the subtree specified by the
     * <code>root</code> node or would be accepted by the filter and
     * <code>whatToShow</code> flags. Further traversal occurs relative to
     * <code>currentNode</code> even if it is not part of the current view,
```

```
     * by applying the filters in the requested direction; if no traversal
     * is possible, <code>currentNode</code> is not changed.
     */
    public Node getCurrentNode();
    /**
     * The node at which the <code>TreeWalker</code> is currently positioned.
     * <br>Alterations to the DOM tree may cause the current node to no longer
     * be accepted by the <code>TreeWalker</code>'s associated filter.
     * <code>currentNode</code> may also be explicitly set to any node,
     * whether or not it is within the subtree specified by the
     * <code>root</code> node or would be accepted by the filter and
     * <code>whatToShow</code> flags. Further traversal occurs relative to
     * <code>currentNode</code> even if it is not part of the current view,
     * by applying the filters in the requested direction; if no traversal
     * is possible, <code>currentNode</code> is not changed.
     * @exception DOMException
     *   NOT_SUPPORTED_ERR: Raised if an attempt is made to set
     *   <code>currentNode</code> to <code>null</code>.
     */
    public void setCurrentNode(Node currentNode)
                        throws DOMException;

    /**
     * Moves to and returns the closest visible ancestor node of the current
     * node. If the search for <code>parentNode</code> attempts to step
     * upward from the <code>TreeWalker</code>'s <code>root</code> node, or
     * if it fails to find a visible ancestor node, this method retains the
     * current position and returns <code>null</code>.
     * @return The new parent node, or <code>null</code> if the current node
     *   has no parent  in the <code>TreeWalker</code>'s logical view.
     */
    public Node parentNode();

    /**
     * Moves the <code>TreeWalker</code> to the first visible child of the
     * current node, and returns the new node. If the current node has no
     * visible children, returns <code>null</code>, and retains the current
     * node.
     * @return The new node, or <code>null</code> if the current node has no
     *   visible children  in the <code>TreeWalker</code>'s logical view.
     */
    public Node firstChild();

    /**
     * Moves the <code>TreeWalker</code> to the last visible child of the
     * current node, and returns the new node. If the current node has no
     * visible children, returns <code>null</code>, and retains the current
     * node.
     * @return The new node, or <code>null</code> if the current node has no
     *   children  in the <code>TreeWalker</code>'s logical view.
     */
    public Node lastChild();

```

```
    /**
     * Moves the <code>TreeWalker</code> to the previous sibling of the
     * current node, and returns the new node. If the current node has no
     * visible previous sibling, returns <code>null</code>, and retains the
     * current node.
     * @return The new node, or <code>null</code> if the current node has no
     *   previous sibling.  in the <code>TreeWalker</code>'s logical view.
     */
    public Node previousSibling();

    /**
     * Moves the <code>TreeWalker</code> to the next sibling of the current
     * node, and returns the new node. If the current node has no visible
     * next sibling, returns <code>null</code>, and retains the current node.
     * @return The new node, or <code>null</code> if the current node has no
     *   next sibling.  in the <code>TreeWalker</code>'s logical view.
     */
    public Node nextSibling();

    /**
     * Moves the <code>TreeWalker</code> to the previous visible node in
     * document order relative to the current node, and returns the new
     * node. If the current node has no previous node,  or if the search for
     * <code>previousNode</code> attempts to step upward from the
     * <code>TreeWalker</code>'s <code>root</code> node,  returns
     * <code>null</code>, and retains the current node.
     * @return The new node, or <code>null</code> if the current node has no
     *   previous node  in the <code>TreeWalker</code>'s logical view.
     */
    public Node previousNode();

    /**
     * Moves the <code>TreeWalker</code> to the next visible node in document
     * order relative to the current node, and returns the new node. If the
     * current node has no next node, or if the search for nextNode attempts
     * to step upward from the <code>TreeWalker</code>'s <code>root</code>
     * node, returns <code>null</code>, and retains the current node.
     * @return The new node, or <code>null</code> if the current node has no
     *   next node  in the <code>TreeWalker</code>'s logical view.
     */
    public Node nextNode();

}
```

45 Dom Views (org.w3c.dom.views package)

45.1 AbstractView.java

Secuencia 470: org/w3c/dom/views/AbstractView.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.views;

/**
 * A base interface that all views shall derive from.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Views-20001113'>Document
     Object Model (DOM) Level 2 Views Specification</a>.
 * @since DOM Level 2
 */
public interface AbstractView {
    /**
     * The source <code>DocumentView</code> of which this is an
     * <code>AbstractView</code>.
     */
    public DocumentView getDocument();

}
```

45.2 DocumentView.java

Secuencia 471: org/w3c/dom/views/DocumentView.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2000 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.views;

/**
 * The <code>DocumentView</code> interface is implemented by
 * <code>Document</code> objects in DOM implementations supporting DOM
 * Views. It provides an attribute to retrieve the default view of a
 * document.
 * <p>See also the <a href='http://www.w3.org/TR/2000/REC-DOM-Level-2-Views-20001113'>Document
     Object Model (DOM) Level 2 Views Specification</a>.
```

```
 * @since DOM Level 2
 */
public interface DocumentView {
    /**
     * The default <code>AbstractView</code> for this <code>Document</code>,
     * or <code>null</code> if none available.
     */
    public AbstractView getDefaultView();

}
```

46 Dom Xpath (org.w3c.dom.xpath package)

46.1 XPathEvaluator.java

Secuencia 472: org/w3c/dom/xpath/XPathEvaluator.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
```

```
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;


import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 *  The evaluation of XPath expressions is provided by
 * <code>XPathEvaluator</code>. In a DOM implementation which supports the
 * XPath 3.0 feature, as described above, the <code>XPathEvaluator</code>
 * interface will be implemented on the same object which implements the
 * <code>Document</code> interface permitting it to be obtained by the usual
 * binding-specific method such as casting or by using the DOM Level 3
 * getInterface method. In this case the implementation obtained from the
 * Document supports the XPath DOM module and is compatible with the XPath
 * 1.0 specification.
 * <p>Evaluation of expressions with specialized extension functions or
 * variables may not work in all implementations and is, therefore, not
 * portable. <code>XPathEvaluator</code> implementations may be available
 * from other sources that could provide specific support for specialized
 * extension functions or variables as would be defined by other
 * specifications.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public interface XPathEvaluator {
    /**
     * Creates a parsed XPath expression with resolved namespaces. This is
     * useful when an expression will be reused in an application since it
     * makes it possible to compile the expression string into a more
     * efficient internal form and preresolve all namespace prefixes which
     * occur within the expression.
     * @param expression The XPath expression string to be parsed.
     * @param resolver The <code>resolver</code> permits translation of
     *   prefixes within the XPath expression into appropriate namespace URIs
     *   . If this is specified as <code>null</code>, any namespace prefix
     *   within the expression will result in <code>DOMException</code>
     *   being thrown with the code <code>NAMESPACE_ERR</code>.
     * @return The compiled form of the XPath expression.
     * @exception XPathException
     *   INVALID_EXPRESSION_ERR: Raised if the expression is not legal
     *   according to the rules of the <code>XPathEvaluator</code>i
     * @exception DOMException
     *   NAMESPACE_ERR: Raised if the expression contains namespace prefixes
     *   which cannot be resolved by the specified
     *   <code>XPathNSResolver</code>.
     */
    public XPathExpression createExpression(String expression,
                                            XPathNSResolver resolver)
```

```
                                        throws XPathException, DOMException;

    /**
     * Adapts any DOM node to resolve namespaces so that an XPath expression
     * can be easily evaluated relative to the context of the node where it
     * appeared within the document. This adapter works like the DOM Level 3
     * method <code>lookupNamespaceURI</code> on nodes in resolving the
     * namespaceURI from a given prefix using the current information
     * available in the node's hierarchy at the time lookupNamespaceURI is
     * called. also correctly resolving the implicit xml prefix.
     * @param nodeResolver The node to be used as a context for namespace
     *   resolution.
     * @return <code>XPathNSResolver</code> which resolves namespaces with
     *   respect to the definitions in scope for a specified node.
     */
    public XPathNSResolver createNSResolver(Node nodeResolver);

    /**
     * Evaluates an XPath expression string and returns a result of the
     * specified type if possible.
     * @param expression The XPath expression string to be parsed and
     *   evaluated.
     * @param contextNode The <code>context</code> is context node for the
     *   evaluation of this XPath expression. If the XPathEvaluator was
     *   obtained by casting the <code>Document</code> then this must be
     *   owned by the same document and must be a <code>Document</code>,
     *   <code>Element</code>, <code>Attribute</code>, <code>Text</code>,
     *   <code>CDATASection</code>, <code>Comment</code>,
     *   <code>ProcessingInstruction</code>, or <code>XPathNamespace</code>
     *   node. If the context node is a <code>Text</code> or a
     *   <code>CDATASection</code>, then the context is interpreted as the
     *   whole logical text node as seen by XPath, unless the node is empty
     *   in which case it may not serve as the XPath context.
     * @param resolver The <code>resolver</code> permits translation of
     *   prefixes within the XPath expression into appropriate namespace URIs
     *   . If this is specified as <code>null</code>, any namespace prefix
     *   within the expression will result in <code>DOMException</code>
     *   being thrown with the code <code>NAMESPACE_ERR</code>.
     * @param type If a specific <code>type</code> is specified, then the
     *   result will be returned as the corresponding type.For XPath 1.0
     *   results, this must be one of the codes of the
     *   <code>XPathResult</code> interface.
     * @param result The <code>result</code> specifies a specific result
     *   object which may be reused and returned by this method. If this is
     *   specified as <code>null</code>or the implementation does not reuse
     *   the specified result, a new result object will be constructed and
     *   returned.For XPath 1.0 results, this object will be of type
     *   <code>XPathResult</code>.
     * @return The result of the evaluation of the XPath expression.For XPath
     *   1.0 results, this object will be of type <code>XPathResult</code>.
     * @exception XPathException
     *   INVALID_EXPRESSION_ERR: Raised if the expression is not legal
     *   according to the rules of the <code>XPathEvaluator</code>i
```

```
     *   <br>TYPE_ERR: Raised if the result cannot be converted to return the
     *   specified type.
     * @exception DOMException
     *   NAMESPACE_ERR: Raised if the expression contains namespace prefixes
     *   which cannot be resolved by the specified
     *   <code>XPathNSResolver</code>.
     *   <br>WRONG_DOCUMENT_ERR: The Node is from a document that is not
     *   supported by this <code>XPathEvaluator</code>.
     *   <br>NOT_SUPPORTED_ERR: The Node is not a type permitted as an XPath
     *   context node or the request type is not permitted by this
     *   <code>XPathEvaluator</code>.
     */
    public Object evaluate(String expression,
                           Node contextNode,
                           XPathNSResolver resolver,
                           short type,
                           Object result)
                           throws XPathException, DOMException;

}
```

46.2 XPathException.java

Secuencia 473: org/w3c/dom/xpath/XPathException.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
```

```
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;

/**
 * A new exception has been created for exceptions specific to these XPath
 * interfaces.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public class XPathException extends RuntimeException {
    public XPathException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
    // XPathExceptionCode
    /**
     * If the expression has a syntax error or otherwise is not a legal
     * expression according to the rules of the specific
     * <code>XPathEvaluator</code> or contains specialized extension
     * functions or variables not supported by this implementation.
     */
    public static final short INVALID_EXPRESSION_ERR    = 1;
    /**
     * If the expression cannot be converted to return the specified type.
     */
    public static final short TYPE_ERR                  = 2;

}
```

46.3 XPathExpression.java

Secuencia 474: org/w3c/dom/xpath/XPathExpression.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;


import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 * The <code>XPathExpression</code> interface represents a parsed and resolved
 * XPath expression.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public interface XPathExpression {
    /**
     * Evaluates this XPath expression and returns a result.
     * @param contextNode The <code>context</code> is context node for the
     *   evaluation of this XPath expression.If the XPathEvaluator was
     *   obtained by casting the <code>Document</code> then this must be
     *   owned by the same document and must be a <code>Document</code>,
     *   <code>Element</code>, <code>Attribute</code>, <code>Text</code>,
     *   <code>CDATASection</code>, <code>Comment</code>,
```

```
     *   <code>ProcessingInstruction</code>, or <code>XPathNamespace</code>
     *   node.If the context node is a <code>Text</code> or a
     *   <code>CDATASection</code>, then the context is interpreted as the
     *   whole logical text node as seen by XPath, unless the node is empty
     *   in which case it may not serve as the XPath context.
     * @param type If a specific <code>type</code> is specified, then the
     *   result will be coerced to return the specified type relying on
     *   XPath conversions and fail if the desired coercion is not possible.
     *   This must be one of the type codes of <code>XPathResult</code>.
     * @param result The <code>result</code> specifies a specific result
     *   object which may be reused and returned by this method. If this is
     *   specified as <code>null</code>or the implementation does not reuse
     *   the specified result, a new result object will be constructed and
     *   returned.For XPath 1.0 results, this object will be of type
     *   <code>XPathResult</code>.
     * @return The result of the evaluation of the XPath expression.For XPath
     *   1.0 results, this object will be of type <code>XPathResult</code>.
     * @exception XPathException
     *   TYPE_ERR: Raised if the result cannot be converted to return the
     *   specified type.
     * @exception DOMException
     *   WRONG_DOCUMENT_ERR: The Node is from a document that is not supported
     *   by the XPathEvaluator that created this <code>XPathExpression</code>
     *   .
     *   <br>NOT_SUPPORTED_ERR: The Node is not a type permitted as an XPath
     *   context node or the request type is not permitted by this
     *   <code>XPathExpression</code>.
     */
    public Object evaluate(Node contextNode,
                           short type,
                           Object result)
                           throws XPathException, DOMException;

}
```

46.4 XPathNSResolver.java

Secuencia 475: org/w3c/dom/xpath/XPathNSResolver.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;


/**
 * The <code>XPathNSResolver</code> interface permit <code>prefix</code>
 * strings in the expression to be properly bound to
 * <code>namespaceURI</code> strings. <code>XPathEvaluator</code> can
 * construct an implementation of <code>XPathNSResolver</code> from a node,
 * or the interface may be implemented by any application.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public interface XPathNSResolver {
    /**
     * Look up the namespace URI associated to the given namespace prefix. The
     * XPath evaluator must never call this with a <code>null</code> or
     * empty argument, because the result of doing this is undefined.
     * @param prefix The prefix to look for.
     * @return Returns the associated namespace URI or <code>null</code> if
     *   none is found.
     */
    public String lookupNamespaceURI(String prefix);

}
```

46.5 XPathNamespace.java

Secuencia 476: org/w3c/dom/xpath/XPathNamespace.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;


import org.w3c.dom.Element;
import org.w3c.dom.Node;

/**
 * The <code>XPathNamespace</code> interface is returned by
 * <code>XPathResult</code> interfaces to represent the XPath namespace node
 * type that DOM lacks. There is no public constructor for this node type.
 * Attempts to place it into a hierarchy or a NamedNodeMap result in a
```

```
 * <code>DOMException</code> with the code <code>HIERARCHY_REQUEST_ERR</code>
 * . This node is read only, so methods or setting of attributes that would
 * mutate the node result in a DOMException with the code
 * <code>NO_MODIFICATION_ALLOWED_ERR</code>.
 * <p>The core specification describes attributes of the <code>Node</code>
 * interface that are different for different node node types but does not
 * describe <code>XPATH_NAMESPACE_NODE</code>, so here is a description of
 * those attributes for this node type. All attributes of <code>Node</code>
 * not described in this section have a <code>null</code> or
 * <code>false</code> value.
 * <p><code>ownerDocument</code> matches the <code>ownerDocument</code> of the
 * <code>ownerElement</code> even if the element is later adopted.
 * <p><code>prefix</code> is the prefix of the namespace represented by the
 * node.
 * <p><code>nodeName</code> is the same as <code>prefix</code>.
 * <p><code>nodeType</code> is equal to <code>XPATH_NAMESPACE_NODE</code>.
 * <p><code>namespaceURI</code> is the namespace URI of the namespace
 * represented by the node.
 * <p><code>adoptNode</code>, <code>cloneNode</code>, and
 * <code>importNode</code> fail on this node type by raising a
 * <code>DOMException</code> with the code <code>NOT_SUPPORTED_ERR</code>.In
 * future versions of the XPath specification, the definition of a namespace
 * node may be changed incomatibly, in which case incompatible changes to
 * field values may be required to implement versions beyond XPath 1.0.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public interface XPathNamespace extends Node {
    // XPathNodeType
    /**
     * The node is a <code>Namespace</code>.
     */
    public static final short XPATH_NAMESPACE_NODE      = 13;

    /**
     * The <code>Element</code> on which the namespace was in scope when it
     * was requested. This does not change on a returned namespace node even
     * if the document changes such that the namespace goes out of scope on
     * that element and this node is no longer found there by XPath.
     */
    public Element getOwnerElement();

}
```

46.6 XPathResult.java

Secuencia 477: org/w3c/dom/xpath/XPathResult.java

```
/*
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/*
 *
 *
 *
 *
 *
 *
 * Copyright (c) 2002 World Wide Web Consortium,
 * (Massachusetts Institute of Technology, Institut National de
 * Recherche en Informatique et en Automatique, Keio University). All
 * Rights Reserved. This program is distributed under the W3C's Software
 * Intellectual Property License. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
 * PURPOSE.
 * See W3C License http://www.w3.org/Consortium/Legal/ for more details.
 */

package org.w3c.dom.xpath;


import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

/**
 * The <code>XPathResult</code> interface represents the result of the
 * evaluation of an XPath 1.0 expression within the context of a particular
 * node. Since evaluation of an XPath expression can result in various
 * result types, this object makes it possible to discover and manipulate
 * the type and value of the result.
 * <p>See also the <a href='http://www.w3.org/2002/08/WD-DOM-Level-3-XPath-20020820'>Document
     Object Model (DOM) Level 3 XPath Specification</a>.
 */
public interface XPathResult {
    // XPathResultType
    /**
```

```
     * This code does not represent a specific type. An evaluation of an XPath
     * expression will never produce this type. If this type is requested,
     * then the evaluation returns whatever type naturally results from
     * evaluation of the expression.
     * <br>If the natural result is a node set when <code>ANY_TYPE</code> was
     * requested, then <code>UNORDERED_NODE_ITERATOR_TYPE</code> is always
     * the resulting type. Any other representation of a node set must be
     * explicitly requested.
     */
    public static final short ANY_TYPE                  = 0;
    /**
     * The result is a number as defined by . Document modification does not
     * invalidate the number, but may mean that reevaluation would not yield
     * the same number.
     */
    public static final short NUMBER_TYPE               = 1;
    /**
     * The result is a string as defined by . Document modification does not
     * invalidate the string, but may mean that the string no longer
     * corresponds to the current document.
     */
    public static final short STRING_TYPE               = 2;
    /**
     * The result is a boolean as defined by . Document modification does not
     * invalidate the boolean, but may mean that reevaluation would not
     * yield the same boolean.
     */
    public static final short BOOLEAN_TYPE              = 3;
    /**
     * The result is a node set as defined by  that will be accessed
     * iteratively, which may not produce nodes in a particular order.
     * Document modification invalidates the iteration.
     * <br>This is the default type returned if the result is a node set and
     * <code>ANY_TYPE</code> is requested.
     */
    public static final short UNORDERED_NODE_ITERATOR_TYPE = 4;
    /**
     * The result is a node set as defined by  that will be accessed
     * iteratively, which will produce document-ordered nodes. Document
     * modification invalidates the iteration.
     */
    public static final short ORDERED_NODE_ITERATOR_TYPE = 5;
    /**
     * The result is a node set as defined by  that will be accessed as a
     * snapshot list of nodes that may not be in a particular order.
     * Document modification does not invalidate the snapshot but may mean
     * that reevaluation would not yield the same snapshot and nodes in the
     * snapshot may have been altered, moved, or removed from the document.
     */
    public static final short UNORDERED_NODE_SNAPSHOT_TYPE = 6;
    /**
     * The result is a node set as defined by  that will be accessed as a
     * snapshot list of nodes that will be in original document order.
```

```
     * Document modification does not invalidate the snapshot but may mean
     * that reevaluation would not yield the same snapshot and nodes in the
     * snapshot may have been altered, moved, or removed from the document.
     */
    public static final short ORDERED_NODE_SNAPSHOT_TYPE = 7;
    /**
     * The result is a node set as defined by  and will be accessed as a
     * single node, which may be <code>null</code>if the node set is empty.
     * Document modification does not invalidate the node, but may mean that
     * the result node no longer corresponds to the current document. This
     * is a convenience that permits optimization since the implementation
     * can stop once any node in the in the resulting set has been found.
     * <br>If there are more than one node in the actual result, the single
     * node returned might not be the first in document order.
     */
    public static final short ANY_UNORDERED_NODE_TYPE   = 8;
    /**
     * The result is a node set as defined by  and will be accessed as a
     * single node, which may be <code>null</code> if the node set is empty.
     * Document modification does not invalidate the node, but may mean that
     * the result node no longer corresponds to the current document. This
     * is a convenience that permits optimization since the implementation
     * can stop once the first node in document order of the resulting set
     * has been found.
     * <br>If there are more than one node in the actual result, the single
     * node returned will be the first in document order.
     */
    public static final short FIRST_ORDERED_NODE_TYPE   = 9;

    /**
     * A code representing the type of this result, as defined by the type
     * constants.
     */
    public short getResultType();

    /**
     * The value of this number result. If the native double type of the DOM
     * binding does not directly support the exact IEEE 754 result of the
     * XPath expression, then it is up to the definition of the binding
     * binding to specify how the XPath number is converted to the native
     * binding number.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>NUMBER_TYPE</code>.
     */
    public double getNumberValue()
                            throws XPathException;

    /**
     * The value of this string result.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>STRING_TYPE</code>.
```

```
     */
    public String getStringValue()
                            throws XPathException;

    /**
     * The value of this boolean result.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>BOOLEAN_TYPE</code>.
     */
    public boolean getBooleanValue()
                            throws XPathException;

    /**
     * The value of this single node result, which may be <code>null</code>.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>ANY_UNORDERED_NODE_TYPE</code> or
     *   <code>FIRST_ORDERED_NODE_TYPE</code>.
     */
    public Node getSingleNodeValue()
                            throws XPathException;

    /**
     * Signifies that the iterator has become invalid. True if
     * <code>resultType</code> is <code>UNORDERED_NODE_ITERATOR_TYPE</code>
     * or <code>ORDERED_NODE_ITERATOR_TYPE</code> and the document has been
     * modified since this result was returned.
     */
    public boolean getInvalidIteratorState();

    /**
     * The number of nodes in the result snapshot. Valid values for
     * snapshotItem indices are <code>0</code> to
     * <code>snapshotLength-1</code> inclusive.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>UNORDERED_NODE_SNAPSHOT_TYPE</code> or
     *   <code>ORDERED_NODE_SNAPSHOT_TYPE</code>.
     */
    public int getSnapshotLength()
                            throws XPathException;

    /**
     * Iterates and returns the next node from the node set or
     * <code>null</code>if there are no more nodes.
     * @return Returns the next node.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>UNORDERED_NODE_ITERATOR_TYPE</code> or
     *   <code>ORDERED_NODE_ITERATOR_TYPE</code>.
     * @exception DOMException
     *   INVALID_STATE_ERR: The document has been mutated since the result was
```

```
     *   returned.
     */
    public Node iterateNext()
                            throws XPathException, DOMException;

    /**
     * Returns the <code>index</code>th item in the snapshot collection. If
     * <code>index</code> is greater than or equal to the number of nodes in
     * the list, this method returns <code>null</code>. Unlike the iterator
     * result, the snapshot does not become invalid, but may not correspond
     * to the current document if it is mutated.
     * @param index Index into the snapshot collection.
     * @return The node at the <code>index</code>th position in the
     *   <code>NodeList</code>, or <code>null</code> if that is not a valid
     *   index.
     * @exception XPathException
     *   TYPE_ERR: raised if <code>resultType</code> is not
     *   <code>UNORDERED_NODE_SNAPSHOT_TYPE</code> or
     *   <code>ORDERED_NODE_SNAPSHOT_TYPE</code>.
     */
    public Node snapshotItem(int index)
                             throws XPathException;

}
```

47 Sax (org.xml.sax package)

47.1 AttributeList.java

Secuencia 478: org/xml/sax/AttributeList.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

// SAX Attribute List Interface.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: AttributeList.java,v 1.3 2004/11/03 22:44:51 jsuttor Exp $

package org.xml.sax;

/**
 * Interface for an element's attribute specifications.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This is the original SAX1 interface for reporting an element's
 * attributes.  Unlike the new {@link org.xml.sax.Attributes Attributes}
 * interface, it does not support Namespace-related information.</p>
 *
 * <p>When an attribute list is supplied as part of a
 * {@link org.xml.sax.DocumentHandler#startElement startElement}
 * event, the list will return valid results only during the
 * scope of the event; once the event handler returns control
 * to the parser, the attribute list is invalid.  To save a
 * persistent copy of the attribute list, use the SAX1
 * {@link org.xml.sax.helpers.AttributeListImpl AttributeListImpl}
 * helper class.</p>
 *
 * <p>An attribute list includes only attributes that have been
 * specified or defaulted: #IMPLIED attributes will not be included.</p>
 *
 * <p>There are two ways for the SAX application to obtain information
 * from the AttributeList.  First, it can iterate through the entire
 * list:</p>
 *
 * <pre>
 * public void startElement (String name, AttributeList atts) {
 *   for (int i = 0; i < atts.getLength(); i++) {
 *     String name = atts.getName(i);
 *     String type = atts.getType(i);
 *     String value = atts.getValue(i);
 *     [...]
 *   }
 * }
 * </pre>
 *
 * <p>(Note that the result of getLength() will be zero if there
 * are no attributes.)
 *
```

```
 * <p>As an alternative, the application can request the value or
 * type of specific attributes:</p>
 *
 * <pre>
 * public void startElement (String name, AttributeList atts) {
 *   String identifier = atts.getValue("id");
 *   String label = atts.getValue("label");
 *   [...]
 * }
 * </pre>
 *
 * @deprecated This interface has been replaced by the SAX2
 *             {@link org.xml.sax.Attributes Attributes}
 *             interface, which includes Namespace support.
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.DocumentHandler#startElement startElement
 * @see org.xml.sax.helpers.AttributeListImpl AttributeListImpl
 */
public interface AttributeList {


    ////////////////////////////////////////////////////////////////////
    // Iteration methods.
    ////////////////////////////////////////////////////////////////////


    /**
     * Return the number of attributes in this list.
     *
     * <p>The SAX parser may provide attributes in any
     * arbitrary order, regardless of the order in which they were
     * declared or specified.  The number of attributes may be
     * zero.</p>
     *
     * @return The number of attributes in the list.
     */
    public abstract int getLength ();


    /**
     * Return the name of an attribute in this list (by position).
     *
     * <p>The names must be unique: the SAX parser shall not include the
     * same attribute twice.  Attributes without values (those declared
     * #IMPLIED without a value specified in the start tag) will be
     * omitted from the list.</p>
     *
     * <p>If the attribute name has a namespace prefix, the prefix
     * will still be attached.</p>
     *
     * @param i The index of the attribute in the list (starting at 0).
     * @return The name of the indexed attribute, or null
```

```
     *         if the index is out of range.
     * @see #getLength
     */
    public abstract String getName (int i);


    /**
     * Return the type of an attribute in the list (by position).
     *
     * <p>The attribute type is one of the strings "CDATA", "ID",
     * "IDREF", "IDREFS", "NMTOKEN", "NMTOKENS", "ENTITY", "ENTITIES",
     * or "NOTATION" (always in upper case).</p>
     *
     * <p>If the parser has not read a declaration for the attribute,
     * or if the parser does not report attribute types, then it must
     * return the value "CDATA" as stated in the XML 1.0 Recommentation
     * (clause 3.3.3, "Attribute-Value Normalization").</p>
     *
     * <p>For an enumerated attribute that is not a notation, the
     * parser will report the type as "NMTOKEN".</p>
     *
     * @param i The index of the attribute in the list (starting at 0).
     * @return The attribute type as a string, or
     *         null if the index is out of range.
     * @see #getLength
     * @see #getType(java.lang.String)
     */
    public abstract String getType (int i);


    /**
     * Return the value of an attribute in the list (by position).
     *
     * <p>If the attribute value is a list of tokens (IDREFS,
     * ENTITIES, or NMTOKENS), the tokens will be concatenated
     * into a single string separated by whitespace.</p>
     *
     * @param i The index of the attribute in the list (starting at 0).
     * @return The attribute value as a string, or
     *         null if the index is out of range.
     * @see #getLength
     * @see #getValue(java.lang.String)
     */
    public abstract String getValue (int i);



    ////////////////////////////////////////////////////////////////////
    // Lookup methods.
    ////////////////////////////////////////////////////////////////////


    /**
```

```
     * Return the type of an attribute in the list (by name).
     *
     * <p>The return value is the same as the return value for
     * getType(int).</p>
     *
     * <p>If the attribute name has a namespace prefix in the document,
     * the application must include the prefix here.</p>
     *
     * @param name The name of the attribute.
     * @return The attribute type as a string, or null if no
     *         such attribute exists.
     * @see #getType(int)
     */
    public abstract String getType (String name);


    /**
     * Return the value of an attribute in the list (by name).
     *
     * <p>The return value is the same as the return value for
     * getValue(int).</p>
     *
     * <p>If the attribute name has a namespace prefix in the document,
     * the application must include the prefix here.</p>
     *
     * @param name the name of the attribute to return
     * @return The attribute value as a string, or null if
     *         no such attribute exists.
     * @see #getValue(int)
     */
    public abstract String getValue (String name);

}

// end of AttributeList.java
```

47.2 Attributes.java

Secuencia 479: org/xml/sax/Attributes.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// Attributes.java - attribute list with Namespace support
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: Attributes.java,v 1.2 2004/11/03 22:44:51 jsuttor Exp $

package org.xml.sax;


/**
 * Interface for a list of XML attributes.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This interface allows access to a list of attributes in
 * three different ways:</p>
 *
 * <ol>
 * <li>by attribute index;</li>
 * <li>by Namespace-qualified name; or</li>
 * <li>by qualified (prefixed) name.</li>
 * </ol>
 *
 * <p>The list will not contain attributes that were declared
 * #IMPLIED but not specified in the start tag.  It will also not
 * contain attributes used as Namespace declarations (xmlns*) unless
 * the <code>http://xml.org/sax/features/namespace-prefixes</code>
 * feature is set to <var>true</var> (it is <var>false</var> by
 * default).
 * Because SAX2 conforms to the original "Namespaces in XML"
 * recommendation, it normally does not
 * give namespace declaration attributes a namespace URI.
 * </p>
 *
 * <p>Some SAX2 parsers may support using an optional feature flag
 * (<code>http://xml.org/sax/features/xmlns-uris</code>) to request
 * that those attributes be given URIs, conforming to a later
```

```
 * backwards-incompatible revision of that recommendation.  (The
 * attribute's "local name" will be the prefix, or "xmlns" when
 * defining a default element namespace.)  For portability, handler
 * code should always resolve that conflict, rather than requiring
 * parsers that can change the setting of that feature flag.  </p>
 *
 * <p>If the namespace-prefixes feature (see above) is
 * <var>false</var>, access by qualified name may not be available; if
 * the <code>http://xml.org/sax/features/namespaces</code> feature is
 * <var>false</var>, access by Namespace-qualified names may not be
 * available.</p>
 *
 * <p>This interface replaces the now-deprecated SAX1 {@link
 * org.xml.sax.AttributeList AttributeList} interface, which does not
 * contain Namespace support.  In addition to Namespace support, it
 * adds the <var>getIndex</var> methods (below).</p>
 *
 * <p>The order of attributes in the list is unspecified, and will
 * vary from implementation to implementation.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.helpers.AttributesImpl
 * @see org.xml.sax.ext.DeclHandler#attributeDecl
 */
public interface Attributes
{


    ////////////////////////////////////////////////////////////////////
    // Indexed access.
    ////////////////////////////////////////////////////////////////////


    /**
     * Return the number of attributes in the list.
     *
     * <p>Once you know the number of attributes, you can iterate
     * through the list.</p>
     *
     * @return The number of attributes in the list.
     * @see #getURI(int)
     * @see #getLocalName(int)
     * @see #getQName(int)
     * @see #getType(int)
     * @see #getValue(int)
     */
    public abstract int getLength ();


    /**
     * Look up an attribute's Namespace URI by index.
     *
```

```
     * @param index The attribute index (zero-based).
     * @return The Namespace URI, or the empty string if none
     *         is available, or null if the index is out of
     *         range.
     * @see #getLength
     */
    public abstract String getURI (int index);


    /**
     * Look up an attribute's local name by index.
     *
     * @param index The attribute index (zero-based).
     * @return The local name, or the empty string if Namespace
     *         processing is not being performed, or null
     *         if the index is out of range.
     * @see #getLength
     */
    public abstract String getLocalName (int index);


    /**
     * Look up an attribute's XML qualified (prefixed) name by index.
     *
     * @param index The attribute index (zero-based).
     * @return The XML qualified name, or the empty string
     *         if none is available, or null if the index
     *         is out of range.
     * @see #getLength
     */
    public abstract String getQName (int index);


    /**
     * Look up an attribute's type by index.
     *
     * <p>The attribute type is one of the strings "CDATA", "ID",
     * "IDREF", "IDREFS", "NMTOKEN", "NMTOKENS", "ENTITY", "ENTITIES",
     * or "NOTATION" (always in upper case).</p>
     *
     * <p>If the parser has not read a declaration for the attribute,
     * or if the parser does not report attribute types, then it must
     * return the value "CDATA" as stated in the XML 1.0 Recommendation
     * (clause 3.3.3, "Attribute-Value Normalization").</p>
     *
     * <p>For an enumerated attribute that is not a notation, the
     * parser will report the type as "NMTOKEN".</p>
     *
     * @param index The attribute index (zero-based).
     * @return The attribute's type as a string, or null if the
     *         index is out of range.
     * @see #getLength
     */
```

```
    public abstract String getType (int index);


    /**
     * Look up an attribute's value by index.
     *
     * <p>If the attribute value is a list of tokens (IDREFS,
     * ENTITIES, or NMTOKENS), the tokens will be concatenated
     * into a single string with each token separated by a
     * single space.</p>
     *
     * @param index The attribute index (zero-based).
     * @return The attribute's value as a string, or null if the
     *         index is out of range.
     * @see #getLength
     */
    public abstract String getValue (int index);



    ////////////////////////////////////////////////////////////////////
    // Name-based query.
    ////////////////////////////////////////////////////////////////////


    /**
     * Look up the index of an attribute by Namespace name.
     *
     * @param uri The Namespace URI, or the empty string if
     *        the name has no Namespace URI.
     * @param localName The attribute's local name.
     * @return The index of the attribute, or -1 if it does not
     *         appear in the list.
     */
    public int getIndex (String uri, String localName);


    /**
     * Look up the index of an attribute by XML qualified (prefixed) name.
     *
     * @param qName The qualified (prefixed) name.
     * @return The index of the attribute, or -1 if it does not
     *         appear in the list.
     */
    public int getIndex (String qName);


    /**
     * Look up an attribute's type by Namespace name.
     *
     * <p>See {@link #getType(int) getType(int)} for a description
     * of the possible types.</p>
     *
```

```
     * @param uri The Namespace URI, or the empty String if the
     *        name has no Namespace URI.
     * @param localName The local name of the attribute.
     * @return The attribute type as a string, or null if the
     *         attribute is not in the list or if Namespace
     *         processing is not being performed.
     */
    public abstract String getType (String uri, String localName);


    /**
     * Look up an attribute's type by XML qualified (prefixed) name.
     *
     * <p>See {@link #getType(int) getType(int)} for a description
     * of the possible types.</p>
     *
     * @param qName The XML qualified name.
     * @return The attribute type as a string, or null if the
     *         attribute is not in the list or if qualified names
     *         are not available.
     */
    public abstract String getType (String qName);


    /**
     * Look up an attribute's value by Namespace name.
     *
     * <p>See {@link #getValue(int) getValue(int)} for a description
     * of the possible values.</p>
     *
     * @param uri The Namespace URI, or the empty String if the
     *        name has no Namespace URI.
     * @param localName The local name of the attribute.
     * @return The attribute value as a string, or null if the
     *         attribute is not in the list.
     */
    public abstract String getValue (String uri, String localName);


    /**
     * Look up an attribute's value by XML qualified (prefixed) name.
     *
     * <p>See {@link #getValue(int) getValue(int)} for a description
     * of the possible values.</p>
     *
     * @param qName The XML qualified name.
     * @return The attribute value as a string, or null if the
     *         attribute is not in the list or if qualified names
     *         are not available.
     */
    public abstract String getValue (String qName);

}
```

```

// end of Attributes.java
```

47.3 ContentHandler.java

Secuencia 480: org/xml/sax/ContentHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// ContentHandler.java - handle main document content.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: ContentHandler.java,v 1.2 2004/11/03 22:44:51 jsuttor Exp $

package org.xml.sax;


/**
 * Receive notification of the logical content of a document.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This is the main interface that most SAX applications
 * implement: if the application needs to be informed of basic parsing
 * events, it implements this interface and registers an instance with
```

```
 * the SAX parser using the {@link org.xml.sax.XMLReader#setContentHandler
 * setContentHandler} method.  The parser uses the instance to report
 * basic document-related events like the start and end of elements
 * and character data.</p>
 *
 * <p>The order of events in this interface is very important, and
 * mirrors the order of information in the document itself.  For
 * example, all of an element's content (character data, processing
 * instructions, and/or subelements) will appear, in order, between
 * the startElement event and the corresponding endElement event.</p>
 *
 * <p>This interface is similar to the now-deprecated SAX 1.0
 * DocumentHandler interface, but it adds support for Namespaces
 * and for reporting skipped entities (in non-validating XML
 * processors).</p>
 *
 * <p>Implementors should note that there is also a
 * <code>ContentHandler</code> class in the <code>java.net</code>
 * package; that means that it's probably a bad idea to do</p>
 *
 * <pre>import java.net.*;
 * import org.xml.sax.*;
 * </pre>
 *
 * <p>In fact, "import ...*" is usually a sign of sloppy programming
 * anyway, so the user should consider this a feature rather than a
 * bug.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.XMLReader
 * @see org.xml.sax.DTDHandler
 * @see org.xml.sax.ErrorHandler
 */
public interface ContentHandler
{

    /**
     * Receive an object for locating the origin of SAX document events.
     *
     * <p>SAX parsers are strongly encouraged (though not absolutely
     * required) to supply a locator: if it does so, it must supply
     * the locator to the application by invoking this method before
     * invoking any of the other methods in the ContentHandler
     * interface.</p>
     *
     * <p>The locator allows the application to determine the end
     * position of any document-related event, even if the parser is
     * not reporting an error.  Typically, the application will
     * use this information for reporting its own errors (such as
     * character content that does not match an application's
     * business rules).  The information returned by the locator
     * is probably not sufficient for use with a search engine.</p>
```

```
     *
     * <p>Note that the locator will return correct information only
     * during the invocation SAX event callbacks after
     * {@link #startDocument startDocument} returns and before
     * {@link #endDocument endDocument} is called.  The
     * application should not attempt to use it at any other time.</p>
     *
     * @param locator an object that can return the location of
     *                any SAX document event
     * @see org.xml.sax.Locator
     */
    public void setDocumentLocator (Locator locator);


    /**
     * Receive notification of the beginning of a document.
     *
     * <p>The SAX parser will invoke this method only once, before any
     * other event callbacks (except for {@link #setDocumentLocator
     * setDocumentLocator}).</p>
     *
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     * @see #endDocument
     */
    public void startDocument ()
        throws SAXException;


    /**
     * Receive notification of the end of a document.
     *
     * <p><strong>There is an apparent contradiction between the
     * documentation for this method and the documentation for {@link
     * org.xml.sax.ErrorHandler#fatalError}.  Until this ambiguity is
     * resolved in a future major release, clients should make no
     * assumptions about whether endDocument() will or will not be
     * invoked when the parser has reported a fatalError() or thrown
     * an exception.</strong></p>
     *
     * <p>The SAX parser will invoke this method only once, and it will
     * be the last method invoked during the parse.  The parser shall
     * not invoke this method until it has either abandoned parsing
     * (because of an unrecoverable error) or reached the end of
     * input.</p>
     *
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     * @see #startDocument
     */
    public void endDocument()
        throws SAXException;

```

```java

    /**
     * Begin the scope of a prefix-URI Namespace mapping.
     *
     * <p>The information from this event is not necessary for
     * normal Namespace processing: the SAX XML reader will
     * automatically replace prefixes for element and attribute
     * names when the <code>http://xml.org/sax/features/namespaces</code>
     * feature is <var>true</var> (the default).</p>
     *
     * <p>There are cases, however, when applications need to
     * use prefixes in character data or in attribute values,
     * where they cannot safely be expanded automatically; the
     * start/endPrefixMapping event supplies the information
     * to the application to expand prefixes in those contexts
     * itself, if necessary.</p>
     *
     * <p>Note that start/endPrefixMapping events are not
     * guaranteed to be properly nested relative to each other:
     * all startPrefixMapping events will occur immediately before the
     * corresponding {@link #startElement startElement} event,
     * and all {@link #endPrefixMapping endPrefixMapping}
     * events will occur immediately after the corresponding
     * {@link #endElement endElement} event,
     * but their order is not otherwise
     * guaranteed.</p>
     *
     * <p>There should never be start/endPrefixMapping events for the
     * "xml" prefix, since it is predeclared and immutable.</p>
     *
     * @param prefix the Namespace prefix being declared.
     *  An empty string is used for the default element namespace,
     *  which has no prefix.
     * @param uri the Namespace URI the prefix is mapped to
     * @throws org.xml.sax.SAXException the client may throw
     *            an exception during processing
     * @see #endPrefixMapping
     * @see #startElement
     */
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException;


    /**
     * End the scope of a prefix-URI mapping.
     *
     * <p>See {@link #startPrefixMapping startPrefixMapping} for
     * details.  These events will always occur immediately after the
     * corresponding {@link #endElement endElement} event, but the order of
     * {@link #endPrefixMapping endPrefixMapping} events is not otherwise
     * guaranteed.</p>
     *
     * @param prefix the prefix that was being mapped.
```

```
     *  This is the empty string when a default mapping scope ends.
     * @throws org.xml.sax.SAXException the client may throw
     *            an exception during processing
     * @see #startPrefixMapping
     * @see #endElement
     */
    public void endPrefixMapping (String prefix)
        throws SAXException;


    /**
     * Receive notification of the beginning of an element.
     *
     * <p>The Parser will invoke this method at the beginning of every
     * element in the XML document; there will be a corresponding
     * {@link #endElement endElement} event for every startElement event
     * (even when the element is empty). All of the element's content will be
     * reported, in order, before the corresponding endElement
     * event.</p>
     *
     * <p>This event allows up to three name components for each
     * element:</p>
     *
     * <ol>
     * <li>the Namespace URI;</li>
     * <li>the local name; and</li>
     * <li>the qualified (prefixed) name.</li>
     * </ol>
     *
     * <p>Any or all of these may be provided, depending on the
     * values of the <var>http://xml.org/sax/features/namespaces</var>
     * and the <var>http://xml.org/sax/features/namespace-prefixes</var>
     * properties:</p>
     *
     * <ul>
     * <li>the Namespace URI and local name are required when
     * the namespaces property is <var>true</var> (the default), and are
     * optional when the namespaces property is <var>false</var> (if one is
     * specified, both must be);</li>
     * <li>the qualified name is required when the namespace-prefixes property
     * is <var>true</var>, and is optional when the namespace-prefixes property
     * is <var>false</var> (the default).</li>
     * </ul>
     *
     * <p>Note that the attribute list provided will contain only
     * attributes with explicit values (specified or defaulted):
     * #IMPLIED attributes will be omitted.  The attribute list
     * will contain attributes used for Namespace declarations
     * (xmlns* attributes) only if the
     * <code>http://xml.org/sax/features/namespace-prefixes</code>
     * property is true (it is false by default, and support for a
     * true value is optional).</p>
     *
```

```
     * <p>Like {@link #characters characters()}, attribute values may have
     * characters that need more than one <code>char</code> value.  </p>
     *
     * @param uri the Namespace URI, or the empty string if the
     *        element has no Namespace URI or if Namespace
     *        processing is not being performed
     * @param localName the local name (without prefix), or the
     *        empty string if Namespace processing is not being
     *        performed
     * @param qName the qualified name (with prefix), or the
     *        empty string if qualified names are not available
     * @param atts the attributes attached to the element.  If
     *        there are no attributes, it shall be an empty
     *        Attributes object.  The value of this object after
     *        startElement returns is undefined
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     * @see #endElement
     * @see org.xml.sax.Attributes
     * @see org.xml.sax.helpers.AttributesImpl
     */
    public void startElement (String uri, String localName,
                              String qName, Attributes atts)
        throws SAXException;


    /**
     * Receive notification of the end of an element.
     *
     * <p>The SAX parser will invoke this method at the end of every
     * element in the XML document; there will be a corresponding
     * {@link #startElement startElement} event for every endElement
     * event (even when the element is empty).</p>
     *
     * <p>For information on the names, see startElement.</p>
     *
     * @param uri the Namespace URI, or the empty string if the
     *        element has no Namespace URI or if Namespace
     *        processing is not being performed
     * @param localName the local name (without prefix), or the
     *        empty string if Namespace processing is not being
     *        performed
     * @param qName the qualified XML name (with prefix), or the
     *        empty string if qualified names are not available
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     */
    public void endElement (String uri, String localName,
                            String qName)
        throws SAXException;


    /**
```

```
     * Receive notification of character data.
     *
     * <p>The Parser will call this method to report each chunk of
     * character data.  SAX parsers may return all contiguous character
     * data in a single chunk, or they may split it into several
     * chunks; however, all of the characters in any single event
     * must come from the same external entity so that the Locator
     * provides useful information.</p>
     *
     * <p>The application must not attempt to read from the array
     * outside of the specified range.</p>
     *
     * <p>Individual characters may consist of more than one Java
     * <code>char</code> value.  There are two important cases where this
     * happens, because characters can't be represented in just sixteen bits.
     * In one case, characters are represented in a <em>Surrogate Pair</em>,
     * using two special Unicode values. Such characters are in the so-called
     * "Astral Planes", with a code point above U+FFFF.  A second case involves
     * composite characters, such as a base character combining with one or
     * more accent characters. </p>
     *
     * <p> Your code should not assume that algorithms using
     * <code>char</code>-at-a-time idioms will be working in character
     * units; in some cases they will split characters.  This is relevant
     * wherever XML permits arbitrary characters, such as attribute values,
     * processing instruction data, and comments as well as in data reported
     * from this method.  It's also generally relevant whenever Java code
     * manipulates internationalized text; the issue isn't unique to XML.</p>
     *
     * <p>Note that some parsers will report whitespace in element
     * content using the {@link #ignorableWhitespace ignorableWhitespace}
     * method rather than this one (validating parsers <em>must</em>
     * do so).</p>
     *
     * @param ch the characters from the XML document
     * @param start the start position in the array
     * @param length the number of characters to read from the array
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     * @see #ignorableWhitespace
     * @see org.xml.sax.Locator
     */
    public void characters (char ch[], int start, int length)
        throws SAXException;


    /**
     * Receive notification of ignorable whitespace in element content.
     *
     * <p>Validating Parsers must use this method to report each chunk
     * of whitespace in element content (see the W3C XML 1.0
     * recommendation, section 2.10): non-validating parsers may also
     * use this method if they are capable of parsing and using
```

```
     * content models.</p>
     *
     * <p>SAX parsers may return all contiguous whitespace in a single
     * chunk, or they may split it into several chunks; however, all of
     * the characters in any single event must come from the same
     * external entity, so that the Locator provides useful
     * information.</p>
     *
     * <p>The application must not attempt to read from the array
     * outside of the specified range.</p>
     *
     * @param ch the characters from the XML document
     * @param start the start position in the array
     * @param length the number of characters to read from the array
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     * @see #characters
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException;


    /**
     * Receive notification of a processing instruction.
     *
     * <p>The Parser will invoke this method once for each processing
     * instruction found: note that processing instructions may occur
     * before or after the main document element.</p>
     *
     * <p>A SAX parser must never report an XML declaration (XML 1.0,
     * section 2.8) or a text declaration (XML 1.0, section 4.3.1)
     * using this method.</p>
     *
     * <p>Like {@link #characters characters()}, processing instruction
     * data may have characters that need more than one <code>char</code>
     * value. </p>
     *
     * @param target the processing instruction target
     * @param data the processing instruction data, or null if
     *        none was supplied.  The data does not include any
     *        whitespace separating it from the target
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     */
    public void processingInstruction (String target, String data)
        throws SAXException;


    /**
     * Receive notification of a skipped entity.
     * This is not called for entity references within markup constructs
     * such as element start tags or markup declarations.  (The XML
     * recommendation requires reporting skipped external entities.
```

```
     * SAX also reports internal entity expansion/non-expansion, except
     * within markup constructs.)
     *
     * <p>The Parser will invoke this method each time the entity is
     * skipped.  Non-validating processors may skip entities if they
     * have not seen the declarations (because, for example, the
     * entity was declared in an external DTD subset).  All processors
     * may skip external entities, depending on the values of the
     * <code>http://xml.org/sax/features/external-general-entities</code>
     * and the
     * <code>http://xml.org/sax/features/external-parameter-entities</code>
     * properties.</p>
     *
     * @param name the name of the skipped entity.  If it is a
     *        parameter entity, the name will begin with '%', and if
     *        it is the external DTD subset, it will be the string
     *        "[dtd]"
     * @throws org.xml.sax.SAXException any SAX exception, possibly
     *            wrapping another exception
     */
    public void skippedEntity (String name)
        throws SAXException;
}

// end of ContentHandler.java
```

47.4 DTDHandler.java

Secuencia 481: org/xml/sax/DTDHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```java

// SAX DTD handler.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: DTDHandler.java,v 1.2 2004/11/03 22:44:51 jsuttor Exp $

package org.xml.sax;

/**
 * Receive notification of basic DTD-related events.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>If a SAX application needs information about notations and
 * unparsed entities, then the application implements this
 * interface and registers an instance with the SAX parser using
 * the parser's setDTDHandler method.  The parser uses the
 * instance to report notation and unparsed entity declarations to
 * the application.</p>
 *
 * <p>Note that this interface includes only those DTD events that
 * the XML recommendation <em>requires</em> processors to report:
 * notation and unparsed entity declarations.</p>
 *
 * <p>The SAX parser may report these events in any order, regardless
 * of the order in which the notations and unparsed entities were
 * declared; however, all DTD events must be reported after the
 * document handler's startDocument event, and before the first
 * startElement event.
 * (If the {@link org.xml.sax.ext.LexicalHandler LexicalHandler} is
 * used, these events must also be reported before the endDTD event.)
 * </p>
 *
 * <p>It is up to the application to store the information for
 * future use (perhaps in a hash table or object tree).
 * If the application encounters attributes of type "NOTATION",
 * "ENTITY", or "ENTITIES", it can use the information that it
 * obtained through this interface to find the entity and/or
 * notation corresponding with the attribute value.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.XMLReader#setDTDHandler
 */
public interface DTDHandler {


    /**
```

```
     * Receive notification of a notation declaration event.
     *
     * <p>It is up to the application to record the notation for later
     * reference, if necessary;
     * notations may appear as attribute values and in unparsed entity
     * declarations, and are sometime used with processing instruction
     * target names.</p>
     *
     * <p>At least one of publicId and systemId must be non-null.
     * If a system identifier is present, and it is a URL, the SAX
     * parser must resolve it fully before passing it to the
     * application through this event.</p>
     *
     * <p>There is no guarantee that the notation declaration will be
     * reported before any unparsed entities that use it.</p>
     *
     * @param name The notation name.
     * @param publicId The notation's public identifier, or null if
     *        none was given.
     * @param systemId The notation's system identifier, or null if
     *        none was given.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see #unparsedEntityDecl
     * @see org.xml.sax.Attributes
     */
    public abstract void notationDecl (String name,
                                       String publicId,
                                       String systemId)
        throws SAXException;


    /**
     * Receive notification of an unparsed entity declaration event.
     *
     * <p>Note that the notation name corresponds to a notation
     * reported by the {@link #notationDecl notationDecl} event.
     * It is up to the application to record the entity for later
     * reference, if necessary;
     * unparsed entities may appear as attribute values.
     * </p>
     *
     * <p>If the system identifier is a URL, the parser must resolve it
     * fully before passing it to the application.</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @param name The unparsed entity's name.
     * @param publicId The entity's public identifier, or null if none
     *        was given.
     * @param systemId The entity's system identifier.
     * @param notationName The name of the associated notation.
     * @see #notationDecl
```

```
     * @see org.xml.sax.Attributes
     */
    public abstract void unparsedEntityDecl (String name,
                                             String publicId,
                                             String systemId,
                                             String notationName)
        throws SAXException;

}

// end of DTDHandler.java
```

47.5 DocumentHandler.java

Secuencia 482: org/xml/sax/DocumentHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX document handler.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: DocumentHandler.java,v 1.2 2004/11/03 22:44:51 jsuttor Exp $

package org.xml.sax;

/**
 * Receive notification of general document events.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
```

```
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This was the main event-handling interface for SAX1; in
 * SAX2, it has been replaced by {@link org.xml.sax.ContentHandler
 * ContentHandler}, which provides Namespace support and reporting
 * of skipped entities.  This interface is included in SAX2 only
 * to support legacy SAX1 applications.</p>
 *
 * <p>The order of events in this interface is very important, and
 * mirrors the order of information in the document itself.  For
 * example, all of an element's content (character data, processing
 * instructions, and/or subelements) will appear, in order, between
 * the startElement event and the corresponding endElement event.</p>
 *
 * <p>Application writers who do not want to implement the entire
 * interface can derive a class from HandlerBase, which implements
 * the default functionality; parser writers can instantiate
 * HandlerBase to obtain a default handler.  The application can find
 * the location of any document event using the Locator interface
 * supplied by the Parser through the setDocumentLocator method.</p>
 *
 * @deprecated This interface has been replaced by the SAX2
 *             {@link org.xml.sax.ContentHandler ContentHandler}
 *             interface, which includes Namespace support.
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.Parser#setDocumentHandler
 * @see org.xml.sax.Locator
 * @see org.xml.sax.HandlerBase
 */
public interface DocumentHandler {


    /**
     * Receive an object for locating the origin of SAX document events.
     *
     * <p>SAX parsers are strongly encouraged (though not absolutely
     * required) to supply a locator: if it does so, it must supply
     * the locator to the application by invoking this method before
     * invoking any of the other methods in the DocumentHandler
     * interface.</p>
     *
     * <p>The locator allows the application to determine the end
     * position of any document-related event, even if the parser is
     * not reporting an error.  Typically, the application will
     * use this information for reporting its own errors (such as
     * character content that does not match an application's
     * business rules).  The information returned by the locator
     * is probably not sufficient for use with a search engine.</p>
     *
     * <p>Note that the locator will return correct information only
```

```
     * during the invocation of the events in this interface.  The
     * application should not attempt to use it at any other time.</p>
     *
     * @param locator An object that can return the location of
     *                any SAX document event.
     * @see org.xml.sax.Locator
     */
    public abstract void setDocumentLocator (Locator locator);


    /**
     * Receive notification of the beginning of a document.
     *
     * <p>The SAX parser will invoke this method only once, before any
     * other methods in this interface or in DTDHandler (except for
     * setDocumentLocator).</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     */
    public abstract void startDocument ()
        throws SAXException;


    /**
     * Receive notification of the end of a document.
     *
     * <p>The SAX parser will invoke this method only once, and it will
     * be the last method invoked during the parse.  The parser shall
     * not invoke this method until it has either abandoned parsing
     * (because of an unrecoverable error) or reached the end of
     * input.</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     */
    public abstract void endDocument ()
        throws SAXException;


    /**
     * Receive notification of the beginning of an element.
     *
     * <p>The Parser will invoke this method at the beginning of every
     * element in the XML document; there will be a corresponding
     * endElement() event for every startElement() event (even when the
     * element is empty). All of the element's content will be
     * reported, in order, before the corresponding endElement()
     * event.</p>
     *
     * <p>If the element name has a namespace prefix, the prefix will
     * still be attached.  Note that the attribute list provided will
     * contain only attributes with explicit values (specified or
```

```
     * defaulted): #IMPLIED attributes will be omitted.</p>
     *
     * @param name The element type name.
     * @param atts The attributes attached to the element, if any.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see #endElement
     * @see org.xml.sax.AttributeList
     */
    public abstract void startElement (String name, AttributeList atts)
        throws SAXException;


    /**
     * Receive notification of the end of an element.
     *
     * <p>The SAX parser will invoke this method at the end of every
     * element in the XML document; there will be a corresponding
     * startElement() event for every endElement() event (even when the
     * element is empty).</p>
     *
     * <p>If the element name has a namespace prefix, the prefix will
     * still be attached to the name.</p>
     *
     * @param name The element type name
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     */
    public abstract void endElement (String name)
        throws SAXException;


    /**
     * Receive notification of character data.
     *
     * <p>The Parser will call this method to report each chunk of
     * character data.  SAX parsers may return all contiguous character
     * data in a single chunk, or they may split it into several
     * chunks; however, all of the characters in any single event
     * must come from the same external entity, so that the Locator
     * provides useful information.</p>
     *
     * <p>The application must not attempt to read from the array
     * outside of the specified range.</p>
     *
     * <p>Note that some parsers will report whitespace using the
     * ignorableWhitespace() method rather than this one (validating
     * parsers must do so).</p>
     *
     * @param ch The characters from the XML document.
     * @param start The start position in the array.
     * @param length The number of characters to read from the array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
```

```
     *            wrapping another exception.
     * @see #ignorableWhitespace
     * @see org.xml.sax.Locator
     */
    public abstract void characters (char ch[], int start, int length)
        throws SAXException;


    /**
     * Receive notification of ignorable whitespace in element content.
     *
     * <p>Validating Parsers must use this method to report each chunk
     * of ignorable whitespace (see the W3C XML 1.0 recommendation,
     * section 2.10): non-validating parsers may also use this method
     * if they are capable of parsing and using content models.</p>
     *
     * <p>SAX parsers may return all contiguous whitespace in a single
     * chunk, or they may split it into several chunks; however, all of
     * the characters in any single event must come from the same
     * external entity, so that the Locator provides useful
     * information.</p>
     *
     * <p>The application must not attempt to read from the array
     * outside of the specified range.</p>
     *
     * @param ch The characters from the XML document.
     * @param start The start position in the array.
     * @param length The number of characters to read from the array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see #characters
     */
    public abstract void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException;


    /**
     * Receive notification of a processing instruction.
     *
     * <p>The Parser will invoke this method once for each processing
     * instruction found: note that processing instructions may occur
     * before or after the main document element.</p>
     *
     * <p>A SAX parser should never report an XML declaration (XML 1.0,
     * section 2.8) or a text declaration (XML 1.0, section 4.3.1)
     * using this method.</p>
     *
     * @param target The processing instruction target.
     * @param data The processing instruction data, or null if
     *        none was supplied.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     */
```

```
    public abstract void processingInstruction (String target, String data)
        throws SAXException;

}

// end of DocumentHandler.java
```

47.6 EntityResolver.java

Secuencia 483: org/xml/sax/EntityResolver.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX entity resolver.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: EntityResolver.java,v 1.2 2004/11/03 22:44:52 jsuttor Exp $

package org.xml.sax;

import java.io.IOException;


/**
 * Basic interface for resolving entities.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
```

```
 * </blockquote>
 *
 * <p>If a SAX application needs to implement customized handling
 * for external entities, it must implement this interface and
 * register an instance with the SAX driver using the
 * {@link org.xml.sax.XMLReader#setEntityResolver setEntityResolver}
 * method.</p>
 *
 * <p>The XML reader will then allow the application to intercept any
 * external entities (including the external DTD subset and external
 * parameter entities, if any) before including them.</p>
 *
 * <p>Many SAX applications will not need to implement this interface,
 * but it will be especially useful for applications that build
 * XML documents from databases or other specialised input sources,
 * or for applications that use URI types other than URLs.</p>
 *
 * <p>The following resolver would provide the application
 * with a special character stream for the entity with the system
 * identifier "http://www.myhost.com/today":</p>
 *
 * <pre>
 * import org.xml.sax.EntityResolver;
 * import org.xml.sax.InputSource;
 *
 * public class MyResolver implements EntityResolver {
 *   public InputSource resolveEntity (String publicId, String systemId)
 *   {
 *     if (systemId.equals("http://www.myhost.com/today")) {
 *             // return a special input source
 *       MyReader reader = new MyReader();
 *       return new InputSource(reader);
 *     } else {
 *             // use the default behaviour
 *       return null;
 *     }
 *   }
 * }
 * </pre>
 *
 * <p>The application can also use this interface to redirect system
 * identifiers to local URIs or to look up replacements in a catalog
 * (possibly by using the public identifier).</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.XMLReader#setEntityResolver
 * @see org.xml.sax.InputSource
 */
public interface EntityResolver {


    /**
```

```
     * Allow the application to resolve external entities.
     *
     * <p>The parser will call this method before opening any external
     * entity except the top-level document entity.  Such entities include
     * the external DTD subset and external parameter entities referenced
     * within the DTD (in either case, only if the parser reads external
     * parameter entities), and external general entities referenced
     * within the document element (if the parser reads external general
     * entities).  The application may request that the parser locate
     * the entity itself, that it use an alternative URI, or that it
     * use data provided by the application (as a character or byte
     * input stream).</p>
     *
     * <p>Application writers can use this method to redirect external
     * system identifiers to secure and/or local URIs, to look up
     * public identifiers in a catalogue, or to read an entity from a
     * database or other input source (including, for example, a dialog
     * box).  Neither XML nor SAX specifies a preferred policy for using
     * public or system IDs to resolve resources.  However, SAX specifies
     * how to interpret any InputSource returned by this method, and that
     * if none is returned, then the system ID will be dereferenced as
     * a URL.  </p>
     *
     * <p>If the system identifier is a URL, the SAX parser must
     * resolve it fully before reporting it to the application.</p>
     *
     * @param publicId The public identifier of the external entity
     *        being referenced, or null if none was supplied.
     * @param systemId The system identifier of the external entity
     *        being referenced.
     * @return An InputSource object describing the new input source,
     *         or null to request that the parser open a regular
     *         URI connection to the system identifier.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException A Java-specific IO exception,
     *            possibly the result of creating a new InputStream
     *            or Reader for the InputSource.
     * @see org.xml.sax.InputSource
     */
    public abstract InputSource resolveEntity (String publicId,
                                               String systemId)
        throws SAXException, IOException;

}

// end of EntityResolver.java
```

47.7 ErrorHandler.java

Secuencia 484: org/xml/sax/ErrorHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
```

```
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX error handler.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: ErrorHandler.java,v 1.2 2004/11/03 22:44:52 jsuttor Exp $

package org.xml.sax;


/**
 * Basic interface for SAX error handlers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>If a SAX application needs to implement customized error
 * handling, it must implement this interface and then register an
 * instance with the XML reader using the
 * {@link org.xml.sax.XMLReader#setErrorHandler setErrorHandler}
 * method.  The parser will then report all errors and warnings
 * through this interface.</p>
 *
 * <p><strong>WARNING:</strong> If an application does <em>not</em>
 * register an ErrorHandler, XML parsing errors will go unreported,
 * except that <em>SAXParseException</em>s will be thrown for fatal errors.
 * In order to detect validity errors, an ErrorHandler that does something
 * with {@link #error error()} calls must be registered.</p>
```

```
 *
 * <p>For XML processing errors, a SAX driver must use this interface
 * in preference to throwing an exception: it is up to the application
 * to decide whether to throw an exception for different types of
 * errors and warnings.  Note, however, that there is no requirement that
 * the parser continue to report additional errors after a call to
 * {@link #fatalError fatalError}.  In other words, a SAX driver class
 * may throw an exception after reporting any fatalError.
 * Also parsers may throw appropriate exceptions for non-XML errors.
 * For example, {@link XMLReader#parse XMLReader.parse()} would throw
 * an IOException for errors accessing entities or the document.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.XMLReader#setErrorHandler
 * @see org.xml.sax.SAXParseException
 */
public interface ErrorHandler {


    /**
     * Receive notification of a warning.
     *
     * <p>SAX parsers will use this method to report conditions that
     * are not errors or fatal errors as defined by the XML
     * recommendation.  The default behaviour is to take no
     * action.</p>
     *
     * <p>The SAX parser must continue to provide normal parsing events
     * after invoking this method: it should still be possible for the
     * application to process the document through to the end.</p>
     *
     * <p>Filters may use this method to report other, non-XML warnings
     * as well.</p>
     *
     * @param exception The warning information encapsulated in a
     *                  SAX parse exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.SAXParseException
     */
    public abstract void warning (SAXParseException exception)
        throws SAXException;


    /**
     * Receive notification of a recoverable error.
     *
     * <p>This corresponds to the definition of "error" in section 1.2
     * of the W3C XML 1.0 Recommendation.  For example, a validating
     * parser would use this callback to report the violation of a
     * validity constraint.  The default behaviour is to take no
     * action.</p>
```

```
     *
     * <p>The SAX parser must continue to provide normal parsing
     * events after invoking this method: it should still be possible
     * for the application to process the document through to the end.
     * If the application cannot do so, then the parser should report
     * a fatal error even if the XML recommendation does not require
     * it to do so.</p>
     *
     * <p>Filters may use this method to report other, non-XML errors
     * as well.</p>
     *
     * @param exception The error information encapsulated in a
     *                  SAX parse exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.SAXParseException
     */
    public abstract void error (SAXParseException exception)
        throws SAXException;


    /**
     * Receive notification of a non-recoverable error.
     *
     * <p><strong>There is an apparent contradiction between the
     * documentation for this method and the documentation for {@link
     * org.xml.sax.ContentHandler#endDocument}.  Until this ambiguity
     * is resolved in a future major release, clients should make no
     * assumptions about whether endDocument() will or will not be
     * invoked when the parser has reported a fatalError() or thrown
     * an exception.</strong></p>
     *
     * <p>This corresponds to the definition of "fatal error" in
     * section 1.2 of the W3C XML 1.0 Recommendation.  For example, a
     * parser would use this callback to report the violation of a
     * well-formedness constraint.</p>
     *
     * <p>The application must assume that the document is unusable
     * after the parser has invoked this method, and should continue
     * (if at all) only for the sake of collecting additional error
     * messages: in fact, SAX parsers are free to stop reporting any
     * other events once this method has been invoked.</p>
     *
     * @param exception The error information encapsulated in a
     *                  SAX parse exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.SAXParseException
     */
    public abstract void fatalError (SAXParseException exception)
        throws SAXException;

}
```

```

// end of ErrorHandler.java
```

47.8 HandlerBase.java

Secuencia 485: org/xml/sax/HandlerBase.java

```
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX default handler base class.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: HandlerBase.java,v 1.2 2005/06/10 03:50:47 jeffsuttor Exp $

package org.xml.sax;

/**
 * Default base class for handlers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class implements the default behaviour for four SAX1
 * interfaces: EntityResolver, DTDHandler, DocumentHandler,
 * and ErrorHandler.  It is now obsolete, but is included in SAX2 to
 * support legacy SAX1 applications.  SAX2 applications should use
 * the {@link org.xml.sax.helpers.DefaultHandler DefaultHandler}
```

```
 * class instead.</p>
 *
 * <p>Application writers can extend this class when they need to
 * implement only part of an interface; parser writers can
 * instantiate this class to provide default handlers when the
 * application has not supplied its own.</p>
 *
 * <p>Note that the use of this class is optional.</p>
 *
 * @deprecated This class works with the deprecated
 *             {@link org.xml.sax.DocumentHandler DocumentHandler}
 *             interface.  It has been replaced by the SAX2
 *             {@link org.xml.sax.helpers.DefaultHandler DefaultHandler}
 *             class.
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.EntityResolver
 * @see org.xml.sax.DTDHandler
 * @see org.xml.sax.DocumentHandler
 * @see org.xml.sax.ErrorHandler
 */
public class HandlerBase
    implements EntityResolver, DTDHandler, DocumentHandler, ErrorHandler
{


    ////////////////////////////////////////////////////////////////////
    // Default implementation of the EntityResolver interface.
    ////////////////////////////////////////////////////////////////////

    /**
     * Resolve an external entity.
     *
     * <p>Always return null, so that the parser will use the system
     * identifier provided in the XML document.  This method implements
     * the SAX default behaviour: application writers can override it
     * in a subclass to do special translations such as catalog lookups
     * or URI redirection.</p>
     *
     * @param publicId The public identifer, or null if none is
     *                 available.
     * @param systemId The system identifier provided in the XML
     *                 document.
     * @return The new input source, or null to require the
     *         default behaviour.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.EntityResolver#resolveEntity
     */
    public InputSource resolveEntity (String publicId, String systemId)
        throws SAXException
    {
        return null;
```

```
    }



    ////////////////////////////////////////////////////////////////////
    // Default implementation of DTDHandler interface.
    ////////////////////////////////////////////////////////////////////


    /**
     * Receive notification of a notation declaration.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass if they wish to keep track of the notations
     * declared in a document.</p>
     *
     * @param name The notation name.
     * @param publicId The notation public identifier, or null if not
     *                 available.
     * @param systemId The notation system identifier.
     * @see org.xml.sax.DTDHandler#notationDecl
     */
    public void notationDecl (String name, String publicId, String systemId)
    {
        // no op
    }


    /**
     * Receive notification of an unparsed entity declaration.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to keep track of the unparsed entities
     * declared in a document.</p>
     *
     * @param name The entity name.
     * @param publicId The entity public identifier, or null if not
     *                 available.
     * @param systemId The entity system identifier.
     * @param notationName The name of the associated notation.
     * @see org.xml.sax.DTDHandler#unparsedEntityDecl
     */
    public void unparsedEntityDecl (String name, String publicId,
                                    String systemId, String notationName)
    {
        // no op
    }



    ////////////////////////////////////////////////////////////////////
    // Default implementation of DocumentHandler interface.
    ////////////////////////////////////////////////////////////////////
```

```


    /**
     * Receive a Locator object for document events.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass if they wish to store the locator for use
     * with other document events.</p>
     *
     * @param locator A locator for all SAX document events.
     * @see org.xml.sax.DocumentHandler#setDocumentLocator
     * @see org.xml.sax.Locator
     */
    public void setDocumentLocator (Locator locator)
    {
        // no op
    }


    /**
     * Receive notification of the beginning of the document.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the beginning
     * of a document (such as allocating the root node of a tree or
     * creating an output file).</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#startDocument
     */
    public void startDocument ()
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the end of the document.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the end
     * of a document (such as finalising a tree or closing an output
     * file).</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#endDocument
     */
    public void endDocument ()
        throws SAXException
    {
```

```
        // no op
    }


    /**
     * Receive notification of the start of an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the start of
     * each element (such as allocating a new tree node or writing
     * output to a file).</p>
     *
     * @param name The element type name.
     * @param attributes The specified or defaulted attributes.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#startElement
     */
    public void startElement (String name, AttributeList attributes)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the end of an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the end of
     * each element (such as finalising a tree node or writing
     * output to a file).</p>
     *
     * @param name the element name
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#endElement
     */
    public void endElement (String name)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of character data inside an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method to take specific actions for each chunk of character data
     * (such as adding the data to a node or buffer, or printing it to
     * a file).</p>
     *
```

```
     * @param ch The characters.
     * @param start The start position in the character array.
     * @param length The number of characters to use from the
     *               character array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#characters
     */
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of ignorable whitespace in element content.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method to take specific actions for each chunk of ignorable
     * whitespace (such as adding data to a node or buffer, or printing
     * it to a file).</p>
     *
     * @param ch The whitespace characters.
     * @param start The start position in the character array.
     * @param length The number of characters to use from the
     *               character array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#ignorableWhitespace
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of a processing instruction.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions for each
     * processing instruction, such as setting status variables or
     * invoking other methods.</p>
     *
     * @param target The processing instruction target.
     * @param data The processing instruction data, or null if
     *             none is supplied.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DocumentHandler#processingInstruction
     */
```

```
    public void processingInstruction (String target, String data)
        throws SAXException
    {
        // no op
    }



    ////////////////////////////////////////////////////////////////////
    // Default implementation of the ErrorHandler interface.
    ////////////////////////////////////////////////////////////////////


    /**
     * Receive notification of a parser warning.
     *
     * <p>The default implementation does nothing.  Application writers
     * may override this method in a subclass to take specific actions
     * for each warning, such as inserting the message in a log file or
     * printing it to the console.</p>
     *
     * @param e The warning information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#warning
     * @see org.xml.sax.SAXParseException
     */
    public void warning (SAXParseException e)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of a recoverable parser error.
     *
     * <p>The default implementation does nothing.  Application writers
     * may override this method in a subclass to take specific actions
     * for each error, such as inserting the message in a log file or
     * printing it to the console.</p>
     *
     * @param e The warning information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#warning
     * @see org.xml.sax.SAXParseException
     */
    public void error (SAXParseException e)
        throws SAXException
    {
        // no op
    }
```

```


    /**
     * Report a fatal XML parsing error.
     *
     * <p>The default implementation throws a SAXParseException.
     * Application writers may override this method in a subclass if
     * they need to take specific actions for each fatal error (such as
     * collecting all of the errors into a single report): in any case,
     * the application must stop all regular processing when this
     * method is invoked, since the document is no longer reliable, and
     * the parser may no longer report parsing events.</p>
     *
     * @param e The error information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#fatalError
     * @see org.xml.sax.SAXParseException
     */
    public void fatalError (SAXParseException e)
        throws SAXException
    {
        throw e;
    }

}

// end of HandlerBase.java
```

47.9 InputSource.java

Secuencia 486: org/xml/sax/InputSource.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

// SAX input source.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: InputSource.java,v 1.2 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

import java.io.Reader;
import java.io.InputStream;

/**
 * A single input source for an XML entity.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class allows a SAX application to encapsulate information
 * about an input source in a single object, which may include
 * a public identifier, a system identifier, a byte stream (possibly
 * with a specified encoding), and/or a character stream.</p>
 *
 * <p>There are two places that the application can deliver an
 * input source to the parser: as the argument to the Parser.parse
 * method, or as the return value of the EntityResolver.resolveEntity
 * method.</p>
 *
 * <p>The SAX parser will use the InputSource object to determine how
 * to read XML input.  If there is a character stream available, the
 * parser will read that stream directly, disregarding any text
 * encoding declaration found in that stream.
 * If there is no character stream, but there is
 * a byte stream, the parser will use that byte stream, using the
 * encoding specified in the InputSource or else (if no encoding is
 * specified) autodetecting the character encoding using an algorithm
 * such as the one in the XML specification.  If neither a character
 * stream nor a
 * byte stream is available, the parser will attempt to open a URI
 * connection to the resource identified by the system
 * identifier.</p>
 *
 * <p>An InputSource object belongs to the application: the SAX parser
 * shall never modify it in any way (it may modify a copy if
 * necessary).  However, standard processing of both byte and
 * character streams is to close them on as part of end-of-parse cleanup,
 * so applications should not attempt to re-use such streams after they
```

```
 * have been handed to a parser.  </p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.XMLReader#parse(org.xml.sax.InputSource)
 * @see org.xml.sax.EntityResolver#resolveEntity
 * @see java.io.InputStream
 * @see java.io.Reader
 */
public class InputSource {

    /**
     * Zero-argument default constructor.
     *
     * @see #setPublicId
     * @see #setSystemId
     * @see #setByteStream
     * @see #setCharacterStream
     * @see #setEncoding
     */
    public InputSource ()
    {
    }


    /**
     * Create a new input source with a system identifier.
     *
     * <p>Applications may use setPublicId to include a
     * public identifier as well, or setEncoding to specify
     * the character encoding, if known.</p>
     *
     * <p>If the system identifier is a URL, it must be fully
     * resolved (it may not be a relative URL).</p>
     *
     * @param systemId The system identifier (URI).
     * @see #setPublicId
     * @see #setSystemId
     * @see #setByteStream
     * @see #setEncoding
     * @see #setCharacterStream
     */
    public InputSource (String systemId)
    {
        setSystemId(systemId);
    }


    /**
     * Create a new input source with a byte stream.
     *
     * <p>Application writers should use setSystemId() to provide a base
     * for resolving relative URIs, may use setPublicId to include a
```

```
     * public identifier, and may use setEncoding to specify the object's
     * character encoding.</p>
     *
     * @param byteStream The raw byte stream containing the document.
     * @see #setPublicId
     * @see #setSystemId
     * @see #setEncoding
     * @see #setByteStream
     * @see #setCharacterStream
     */
    public InputSource (InputStream byteStream)
    {
        setByteStream(byteStream);
    }


    /**
     * Create a new input source with a character stream.
     *
     * <p>Application writers should use setSystemId() to provide a base
     * for resolving relative URIs, and may use setPublicId to include a
     * public identifier.</p>
     *
     * <p>The character stream shall not include a byte order mark.</p>
     *
     * @see #setPublicId
     * @see #setSystemId
     * @see #setByteStream
     * @see #setCharacterStream
     */
    public InputSource (Reader characterStream)
    {
        setCharacterStream(characterStream);
    }


    /**
     * Set the public identifier for this input source.
     *
     * <p>The public identifier is always optional: if the application
     * writer includes one, it will be provided as part of the
     * location information.</p>
     *
     * @param publicId The public identifier as a string.
     * @see #getPublicId
     * @see org.xml.sax.Locator#getPublicId
     * @see org.xml.sax.SAXParseException#getPublicId
     */
    public void setPublicId (String publicId)
    {
        this.publicId = publicId;
    }

```

```java

    /**
     * Get the public identifier for this input source.
     *
     * @return The public identifier, or null if none was supplied.
     * @see #setPublicId
     */
    public String getPublicId ()
    {
        return publicId;
    }


    /**
     * Set the system identifier for this input source.
     *
     * <p>The system identifier is optional if there is a byte stream
     * or a character stream, but it is still useful to provide one,
     * since the application can use it to resolve relative URIs
     * and can include it in error messages and warnings (the parser
     * will attempt to open a connection to the URI only if
     * there is no byte stream or character stream specified).</p>
     *
     * <p>If the application knows the character encoding of the
     * object pointed to by the system identifier, it can register
     * the encoding using the setEncoding method.</p>
     *
     * <p>If the system identifier is a URL, it must be fully
     * resolved (it may not be a relative URL).</p>
     *
     * @param systemId The system identifier as a string.
     * @see #setEncoding
     * @see #getSystemId
     * @see org.xml.sax.Locator#getSystemId
     * @see org.xml.sax.SAXParseException#getSystemId
     */
    public void setSystemId (String systemId)
    {
        this.systemId = systemId;
    }


    /**
     * Get the system identifier for this input source.
     *
     * <p>The getEncoding method will return the character encoding
     * of the object pointed to, or null if unknown.</p>
     *
     * <p>If the system ID is a URL, it will be fully resolved.</p>
     *
     * @return The system identifier, or null if none was supplied.
     * @see #setSystemId
     * @see #getEncoding
```

```
     */
    public String getSystemId ()
    {
        return systemId;
    }


    /**
     * Set the byte stream for this input source.
     *
     * <p>The SAX parser will ignore this if there is also a character
     * stream specified, but it will use a byte stream in preference
     * to opening a URI connection itself.</p>
     *
     * <p>If the application knows the character encoding of the
     * byte stream, it should set it with the setEncoding method.</p>
     *
     * @param byteStream A byte stream containing an XML document or
     *        other entity.
     * @see #setEncoding
     * @see #getByteStream
     * @see #getEncoding
     * @see java.io.InputStream
     */
    public void setByteStream (InputStream byteStream)
    {
        this.byteStream = byteStream;
    }


    /**
     * Get the byte stream for this input source.
     *
     * <p>The getEncoding method will return the character
     * encoding for this byte stream, or null if unknown.</p>
     *
     * @return The byte stream, or null if none was supplied.
     * @see #getEncoding
     * @see #setByteStream
     */
    public InputStream getByteStream ()
    {
        return byteStream;
    }


    /**
     * Set the character encoding, if known.
     *
     * <p>The encoding must be a string acceptable for an
     * XML encoding declaration (see section 4.3.3 of the XML 1.0
     * recommendation).</p>
     *
```

```
     * <p>This method has no effect when the application provides a
     * character stream.</p>
     *
     * @param encoding A string describing the character encoding.
     * @see #setSystemId
     * @see #setByteStream
     * @see #getEncoding
     */
    public void setEncoding (String encoding)
    {
        this.encoding = encoding;
    }


    /**
     * Get the character encoding for a byte stream or URI.
     * This value will be ignored when the application provides a
     * character stream.
     *
     * @return The encoding, or null if none was supplied.
     * @see #setByteStream
     * @see #getSystemId
     * @see #getByteStream
     */
    public String getEncoding ()
    {
        return encoding;
    }


    /**
     * Set the character stream for this input source.
     *
     * <p>If there is a character stream specified, the SAX parser
     * will ignore any byte stream and will not attempt to open
     * a URI connection to the system identifier.</p>
     *
     * @param characterStream The character stream containing the
     *        XML document or other entity.
     * @see #getCharacterStream
     * @see java.io.Reader
     */
    public void setCharacterStream (Reader characterStream)
    {
        this.characterStream = characterStream;
    }


    /**
     * Get the character stream for this input source.
     *
     * @return The character stream, or null if none was supplied.
     * @see #setCharacterStream
```

```
     */
    public Reader getCharacterStream ()
    {
        return characterStream;
    }



    ////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////

    private String publicId;
    private String systemId;
    private InputStream byteStream;
    private String encoding;
    private Reader characterStream;

}

// end of InputSource.java
```

47.10 Locator.java

Secuencia 487: org/xml/sax/Locator.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX locator interface for document events.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
```

```
// $Id: Locator.java,v 1.2 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;


/**
 * Interface for associating a SAX event with a document location.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>If a SAX parser provides location information to the SAX
 * application, it does so by implementing this interface and then
 * passing an instance to the application using the content
 * handler's {@link org.xml.sax.ContentHandler#setDocumentLocator
 * setDocumentLocator} method.  The application can use the
 * object to obtain the location of any other SAX event
 * in the XML source document.</p>
 *
 * <p>Note that the results returned by the object will be valid only
 * during the scope of each callback method: the application
 * will receive unpredictable results if it attempts to use the
 * locator at any other time, or after parsing completes.</p>
 *
 * <p>SAX parsers are not required to supply a locator, but they are
 * very strongly encouraged to do so.  If the parser supplies a
 * locator, it must do so before reporting any other document events.
 * If no locator has been set by the time the application receives
 * the {@link org.xml.sax.ContentHandler#startDocument startDocument}
 * event, the application should assume that a locator is not
 * available.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.ContentHandler#setDocumentLocator
 */
public interface Locator {


    /**
     * Return the public identifier for the current document event.
     *
     * <p>The return value is the public identifier of the document
     * entity or of the external parsed entity in which the markup
     * triggering the event appears.</p>
     *
     * @return A string containing the public identifier, or
     *         null if none is available.
     * @see #getSystemId
```

```
     */
    public abstract String getPublicId ();


    /**
     * Return the system identifier for the current document event.
     *
     * <p>The return value is the system identifier of the document
     * entity or of the external parsed entity in which the markup
     * triggering the event appears.</p>
     *
     * <p>If the system identifier is a URL, the parser must resolve it
     * fully before passing it to the application.  For example, a file
     * name must always be provided as a <em>file:...</em> URL, and other
     * kinds of relative URI are also resolved against their bases.</p>
     *
     * @return A string containing the system identifier, or null
     *         if none is available.
     * @see #getPublicId
     */
    public abstract String getSystemId ();


    /**
     * Return the line number where the current document event ends.
     * Lines are delimited by line ends, which are defined in
     * the XML specification.
     *
     * <p><strong>Warning:</strong> The return value from the method
     * is intended only as an approximation for the sake of diagnostics;
     * it is not intended to provide sufficient information
     * to edit the character content of the original XML document.
     * In some cases, these "line" numbers match what would be displayed
     * as columns, and in others they may not match the source text
     * due to internal entity expansion.  </p>
     *
     * <p>The return value is an approximation of the line number
     * in the document entity or external parsed entity where the
     * markup triggering the event appears.</p>
     *
     * <p>If possible, the SAX driver should provide the line position
     * of the first character after the text associated with the document
     * event.  The first line is line 1.</p>
     *
     * @return The line number, or -1 if none is available.
     * @see #getColumnNumber
     */
    public abstract int getLineNumber ();


    /**
     * Return the column number where the current document event ends.
     * This is one-based number of Java <code>char</code> values since
```

```
     * the last line end.
     *
     * <p><strong>Warning:</strong> The return value from the method
     * is intended only as an approximation for the sake of diagnostics;
     * it is not intended to provide sufficient information
     * to edit the character content of the original XML document.
     * For example, when lines contain combining character sequences, wide
     * characters, surrogate pairs, or bi-directional text, the value may
     * not correspond to the column in a text editor's display. </p>
     *
     * <p>The return value is an approximation of the column number
     * in the document entity or external parsed entity where the
     * markup triggering the event appears.</p>
     *
     * <p>If possible, the SAX driver should provide the line position
     * of the first character after the text associated with the document
     * event.  The first column in each line is column 1.</p>
     *
     * @return The column number, or -1 if none is available.
     * @see #getLineNumber
     */
    public abstract int getColumnNumber ();

}

// end of Locator.java
```

47.11 Parser.java

Secuencia 488: org/xml/sax/Parser.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 */

// SAX parser interface.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: Parser.java,v 1.2 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

import java.io.IOException;
import java.util.Locale;


/**
 * Basic interface for SAX (Simple API for XML) parsers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This was the main event supplier interface for SAX1; it has
 * been replaced in SAX2 by {@link org.xml.sax.XMLReader XMLReader},
 * which includes Namespace support and sophisticated configurability
 * and extensibility.</p>
 *
 * <p>All SAX1 parsers must implement this basic interface: it allows
 * applications to register handlers for different types of events
 * and to initiate a parse from a URI, or a character stream.</p>
 *
 * <p>All SAX1 parsers must also implement a zero-argument constructor
 * (though other constructors are also allowed).</p>
 *
 * <p>SAX1 parsers are reusable but not re-entrant: the application
 * may reuse a parser object (possibly with a different input source)
 * once the first parse has completed successfully, but it may not
 * invoke the parse() methods recursively within a parse.</p>
 *
 * @deprecated This interface has been replaced by the SAX2
 *             {@link org.xml.sax.XMLReader XMLReader}
 *             interface, which includes Namespace support.
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.EntityResolver
 * @see org.xml.sax.DTDHandler
 * @see org.xml.sax.DocumentHandler
 * @see org.xml.sax.ErrorHandler
 * @see org.xml.sax.HandlerBase
 * @see org.xml.sax.InputSource
 */
public interface Parser
```

```
{

    /**
     * Allow an application to request a locale for errors and warnings.
     *
     * <p>SAX parsers are not required to provide localisation for errors
     * and warnings; if they cannot support the requested locale,
     * however, they must throw a SAX exception.  Applications may
     * not request a locale change in the middle of a parse.</p>
     *
     * @param locale A Java Locale object.
     * @exception org.xml.sax.SAXException Throws an exception
     *            (using the previous or default locale) if the
     *            requested locale is not supported.
     * @see org.xml.sax.SAXException
     * @see org.xml.sax.SAXParseException
     */
    public abstract void setLocale (Locale locale)
        throws SAXException;


    /**
     * Allow an application to register a custom entity resolver.
     *
     * <p>If the application does not register an entity resolver, the
     * SAX parser will resolve system identifiers and open connections
     * to entities itself (this is the default behaviour implemented in
     * HandlerBase).</p>
     *
     * <p>Applications may register a new or different entity resolver
     * in the middle of a parse, and the SAX parser must begin using
     * the new resolver immediately.</p>
     *
     * @param resolver The object for resolving entities.
     * @see EntityResolver
     * @see HandlerBase
     */
    public abstract void setEntityResolver (EntityResolver resolver);


    /**
     * Allow an application to register a DTD event handler.
     *
     * <p>If the application does not register a DTD handler, all DTD
     * events reported by the SAX parser will be silently
     * ignored (this is the default behaviour implemented by
     * HandlerBase).</p>
     *
     * <p>Applications may register a new or different
     * handler in the middle of a parse, and the SAX parser must
     * begin using the new handler immediately.</p>
     *
     * @param handler The DTD handler.
```

```
     * @see DTDHandler
     * @see HandlerBase
     */
    public abstract void setDTDHandler (DTDHandler handler);


    /**
     * Allow an application to register a document event handler.
     *
     * <p>If the application does not register a document handler, all
     * document events reported by the SAX parser will be silently
     * ignored (this is the default behaviour implemented by
     * HandlerBase).</p>
     *
     * <p>Applications may register a new or different handler in the
     * middle of a parse, and the SAX parser must begin using the new
     * handler immediately.</p>
     *
     * @param handler The document handler.
     * @see DocumentHandler
     * @see HandlerBase
     */
    public abstract void setDocumentHandler (DocumentHandler handler);


    /**
     * Allow an application to register an error event handler.
     *
     * <p>If the application does not register an error event handler,
     * all error events reported by the SAX parser will be silently
     * ignored, except for fatalError, which will throw a SAXException
     * (this is the default behaviour implemented by HandlerBase).</p>
     *
     * <p>Applications may register a new or different handler in the
     * middle of a parse, and the SAX parser must begin using the new
     * handler immediately.</p>
     *
     * @param handler The error handler.
     * @see ErrorHandler
     * @see SAXException
     * @see HandlerBase
     */
    public abstract void setErrorHandler (ErrorHandler handler);


    /**
     * Parse an XML document.
     *
     * <p>The application can use this method to instruct the SAX parser
     * to begin parsing an XML document from any valid input
     * source (a character stream, a byte stream, or a URI).</p>
     *
     * <p>Applications may not invoke this method while a parse is in
```

```
     * progress (they should create a new Parser instead for each
     * additional XML document).  Once a parse is complete, an
     * application may reuse the same Parser object, possibly with a
     * different input source.</p>
     *
     * @param source The input source for the top-level of the
     *        XML document.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     * @see org.xml.sax.InputSource
     * @see #parse(java.lang.String)
     * @see #setEntityResolver
     * @see #setDTDHandler
     * @see #setDocumentHandler
     * @see #setErrorHandler
     */
    public abstract void parse (InputSource source)
        throws SAXException, IOException;


    /**
     * Parse an XML document from a system identifier (URI).
     *
     * <p>This method is a shortcut for the common case of reading a
     * document from a system identifier.  It is the exact
     * equivalent of the following:</p>
     *
     * <pre>
     * parse(new InputSource(systemId));
     * </pre>
     *
     * <p>If the system identifier is a URL, it must be fully resolved
     * by the application before it is passed to the parser.</p>
     *
     * @param systemId The system identifier (URI).
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     * @see #parse(org.xml.sax.InputSource)
     */
    public abstract void parse (String systemId)
        throws SAXException, IOException;

}

// end of Parser.java
```

47.12 SAXException.java

Secuencia 489: org/xml/sax/SAXException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX exception class.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: SAXException.java,v 1.3 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

/**
 * Encapsulate a general SAX error or warning.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class can contain basic error or warning information from
 * either the XML parser or the application: a parser writer or
 * application writer can subclass it to provide additional
 * functionality.  SAX handlers may throw this exception or
 * any exception subclassed from it.</p>
 *
 * <p>If the application needs to pass through other types of
 * exceptions, it must wrap those exceptions in a SAXException
```

```
 * or an exception derived from a SAXException.</p>
 *
 * <p>If the parser or application needs to include information about a
 * specific location in an XML document, it should use the
 * {@link org.xml.sax.SAXParseException SAXParseException} subclass.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @version 2.0.1 (sax2r2)
 * @see org.xml.sax.SAXParseException
 */
public class SAXException extends Exception {


    /**
     * Create a new SAXException.
     */
    public SAXException ()
    {
        super();
        this.exception = null;
    }


    /**
     * Create a new SAXException.
     *
     * @param message The error or warning message.
     */
    public SAXException (String message) {
        super(message);
        this.exception = null;
    }


    /**
     * Create a new SAXException wrapping an existing exception.
     *
     * <p>The existing exception will be embedded in the new
     * one, and its message will become the default message for
     * the SAXException.</p>
     *
     * @param e The exception to be wrapped in a SAXException.
     */
    public SAXException (Exception e)
    {
        super();
        this.exception = e;
    }


    /**
     * Create a new SAXException from an existing exception.
```

```
     *
     * <p>The existing exception will be embedded in the new
     * one, but the new exception will have its own message.</p>
     *
     * @param message The detail message.
     * @param e The exception to be wrapped in a SAXException.
     */
    public SAXException (String message, Exception e)
    {
        super(message);
        this.exception = e;
    }


    /**
     * Return a detail message for this exception.
     *
     * <p>If there is an embedded exception, and if the SAXException
     * has no detail message of its own, this method will return
     * the detail message from the embedded exception.</p>
     *
     * @return The error or warning message.
     */
    public String getMessage ()
    {
        String message = super.getMessage();

        if (message == null && exception != null) {
            return exception.getMessage();
        } else {
            return message;
        }
    }


    /**
     * Return the embedded exception, if any.
     *
     * @return The embedded exception, or null if there is none.
     */
    public Exception getException ()
    {
        return exception;
    }

    /**
     * Return the cause of the exception
     *
     * @return Return the cause of the exception
     */
    public Throwable getCause() {
        return exception;
    }
```

```

    /**
     * Override toString to pick up any embedded exception.
     *
     * @return A string representation of this exception.
     */
    public String toString ()
    {
        if (exception != null) {
            return super.toString() + "\n" + exception.toString();
        } else {
            return super.toString();
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////


    /**
     * @serial The embedded exception if tunnelling, or null.
     */
    private Exception exception;

    // Added serialVersionUID to preserve binary compatibility
    static final long serialVersionUID = 583241635256073760L;
}

// end of SAXException.java
```

47.13 SAXNotRecognizedException.java

Secuencia 490: org/xml/sax/SAXNotRecognizedException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 */

// SAXNotRecognizedException.java - unrecognized feature or value.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the Public Domain.
// $Id: SAXNotRecognizedException.java,v 1.3 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;


/**
 * Exception class for an unrecognized identifier.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>An XMLReader will throw this exception when it finds an
 * unrecognized feature or property identifier; SAX applications and
 * extensions may use this class for other, similar purposes.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.SAXNotSupportedException
 */
public class SAXNotRecognizedException extends SAXException
{

    /**
     * Default constructor.
     */
    public SAXNotRecognizedException ()
    {
        super();
    }


    /**
     * Construct a new exception with the given message.
     *
     * @param message The text message of the exception.
     */
    public SAXNotRecognizedException (String message)
```

```
    {
        super(message);
    }

    // Added serialVersionUID to preserve binary compatibility
    static final long serialVersionUID = 5440506620509557213L;
}

// end of SAXNotRecognizedException.java
```

47.14 SAXNotSupportedException.java

Secuencia 491: org/xml/sax/SAXNotSupportedException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAXNotSupportedException.java - unsupported feature or value.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the Public Domain.
// $Id: SAXNotSupportedException.java,v 1.4 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

/**
 * Exception class for an unsupported operation.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
```

```
 * for further information.
 * </blockquote>
 *
 * <p>An XMLReader will throw this exception when it recognizes a
 * feature or property identifier, but cannot perform the requested
 * operation (setting a state or value).  Other SAX2 applications and
 * extensions may use this class for similar purposes.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.SAXNotRecognizedException
 */
public class SAXNotSupportedException extends SAXException
{

    /**
     * Construct a new exception with no message.
     */
    public SAXNotSupportedException ()
    {
        super();
    }


    /**
     * Construct a new exception with the given message.
     *
     * @param message The text message of the exception.
     */
    public SAXNotSupportedException (String message)
    {
        super(message);
    }

    // Added serialVersionUID to preserve binary compatibility
    static final long serialVersionUID = -1422818934641823846L;
}

// end of SAXNotSupportedException.java
```

47.15 SAXParseException.java

Secuencia 492: org/xml/sax/SAXParseException.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX exception class.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: SAXParseException.java,v 1.2 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

/**
 * Encapsulate an XML parse error or warning.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This exception may include information for locating the error
 * in the original XML document, as if it came from a {@link Locator}
 * object.  Note that although the application
 * will receive a SAXParseException as the argument to the handlers
 * in the {@link org.xml.sax.ErrorHandler ErrorHandler} interface,
 * the application is not actually required to throw the exception;
 * instead, it can simply read the information in it and take a
 * different action.</p>
 *
 * <p>Since this exception is a subclass of {@link org.xml.sax.SAXException
 * SAXException}, it inherits the ability to wrap another exception.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @version 2.0.1 (sax2r2)
 * @see org.xml.sax.SAXException
 * @see org.xml.sax.Locator
 * @see org.xml.sax.ErrorHandler
 */
public class SAXParseException extends SAXException {

```

```java

    ////////////////////////////////////////////////////////////////////
    // Constructors.
    ////////////////////////////////////////////////////////////////////


    /**
     * Create a new SAXParseException from a message and a Locator.
     *
     * <p>This constructor is especially useful when an application is
     * creating its own exception from within a {@link org.xml.sax.ContentHandler
     * ContentHandler} callback.</p>
     *
     * @param message The error or warning message.
     * @param locator The locator object for the error or warning (may be
     *        null).
     * @see org.xml.sax.Locator
     */
    public SAXParseException (String message, Locator locator) {
        super(message);
        if (locator != null) {
            init(locator.getPublicId(), locator.getSystemId(),
                 locator.getLineNumber(), locator.getColumnNumber());
        } else {
            init(null, null, -1, -1);
        }
    }


    /**
     * Wrap an existing exception in a SAXParseException.
     *
     * <p>This constructor is especially useful when an application is
     * creating its own exception from within a {@link org.xml.sax.ContentHandler
     * ContentHandler} callback, and needs to wrap an existing exception that is not a
     * subclass of {@link org.xml.sax.SAXException SAXException}.</p>
     *
     * @param message The error or warning message, or null to
     *                use the message from the embedded exception.
     * @param locator The locator object for the error or warning (may be
     *        null).
     * @param e Any exception.
     * @see org.xml.sax.Locator
     */
    public SAXParseException (String message, Locator locator,
                              Exception e) {
        super(message, e);
        if (locator != null) {
            init(locator.getPublicId(), locator.getSystemId(),
                 locator.getLineNumber(), locator.getColumnNumber());
        } else {
            init(null, null, -1, -1);
        }
```

```
    }


    /**
     * Create a new SAXParseException.
     *
     * <p>This constructor is most useful for parser writers.</p>
     *
     * <p>All parameters except the message are as if
     * they were provided by a {@link Locator}.  For example, if the
     * system identifier is a URL (including relative filename), the
     * caller must resolve it fully before creating the exception.</p>
     *
     *
     * @param message The error or warning message.
     * @param publicId The public identifier of the entity that generated
     *                 the error or warning.
     * @param systemId The system identifier of the entity that generated
     *                 the error or warning.
     * @param lineNumber The line number of the end of the text that
     *                   caused the error or warning.
     * @param columnNumber The column number of the end of the text that
     *                     cause the error or warning.
     */
    public SAXParseException (String message, String publicId, String systemId,
                              int lineNumber, int columnNumber)
    {
        super(message);
        init(publicId, systemId, lineNumber, columnNumber);
    }


    /**
     * Create a new SAXParseException with an embedded exception.
     *
     * <p>This constructor is most useful for parser writers who
     * need to wrap an exception that is not a subclass of
     * {@link org.xml.sax.SAXException SAXException}.</p>
     *
     * <p>All parameters except the message and exception are as if
     * they were provided by a {@link Locator}.  For example, if the
     * system identifier is a URL (including relative filename), the
     * caller must resolve it fully before creating the exception.</p>
     *
     * @param message The error or warning message, or null to use
     *                the message from the embedded exception.
     * @param publicId The public identifier of the entity that generated
     *                 the error or warning.
     * @param systemId The system identifier of the entity that generated
     *                 the error or warning.
     * @param lineNumber The line number of the end of the text that
     *                   caused the error or warning.
     * @param columnNumber The column number of the end of the text that
```

```
     *                     cause the error or warning.
     * @param e Another exception to embed in this one.
     */
    public SAXParseException (String message, String publicId, String systemId,
                              int lineNumber, int columnNumber, Exception e)
    {
        super(message, e);
        init(publicId, systemId, lineNumber, columnNumber);
    }


    /**
     * Internal initialization method.
     *
     * @param publicId The public identifier of the entity which generated the exception,
     *        or null.
     * @param systemId The system identifier of the entity which generated the exception,
     *        or null.
     * @param lineNumber The line number of the error, or -1.
     * @param columnNumber The column number of the error, or -1.
     */
    private void init (String publicId, String systemId,
                       int lineNumber, int columnNumber)
    {
        this.publicId = publicId;
        this.systemId = systemId;
        this.lineNumber = lineNumber;
        this.columnNumber = columnNumber;
    }


    /**
     * Get the public identifier of the entity where the exception occurred.
     *
     * @return A string containing the public identifier, or null
     *         if none is available.
     * @see org.xml.sax.Locator#getPublicId
     */
    public String getPublicId ()
    {
        return this.publicId;
    }


    /**
     * Get the system identifier of the entity where the exception occurred.
     *
     * <p>If the system identifier is a URL, it will have been resolved
     * fully.</p>
     *
     * @return A string containing the system identifier, or null
     *         if none is available.
     * @see org.xml.sax.Locator#getSystemId
```

```
     */
    public String getSystemId ()
    {
        return this.systemId;
    }


    /**
     * The line number of the end of the text where the exception occurred.
     *
     * <p>The first line is line 1.</p>
     *
     * @return An integer representing the line number, or -1
     *         if none is available.
     * @see org.xml.sax.Locator#getLineNumber
     */
    public int getLineNumber ()
    {
        return this.lineNumber;
    }


    /**
     * The column number of the end of the text where the exception occurred.
     *
     * <p>The first column in a line is position 1.</p>
     *
     * @return An integer representing the column number, or -1
     *         if none is available.
     * @see org.xml.sax.Locator#getColumnNumber
     */
    public int getColumnNumber ()
    {
        return this.columnNumber;
    }

    /**
     * Override toString to provide more detailed error message.
     *
     * @return A string representation of this exception.
     */
    public String toString() {
        StringBuilder buf = new StringBuilder(getClass().getName());
        String message = getLocalizedMessage();
        if (publicId!=null)    buf.append("publicId: ").append(publicId);
        if (systemId!=null)    buf.append("; systemId: ").append(systemId);
        if (lineNumber!=-1)    buf.append("; lineNumber: ").append(lineNumber);
        if (columnNumber!=-1)  buf.append("; columnNumber: ").append(columnNumber);

       //append the exception message at the end
        if (message!=null)     buf.append("; ").append(message);
        return buf.toString();
    }
```

```java

    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////


    /**
     * @serial The public identifier, or null.
     * @see #getPublicId
     */
    private String publicId;


    /**
     * @serial The system identifier, or null.
     * @see #getSystemId
     */
    private String systemId;


    /**
     * @serial The line number, or -1.
     * @see #getLineNumber
     */
    private int lineNumber;


    /**
     * @serial The column number, or -1.
     * @see #getColumnNumber
     */
    private int columnNumber;

    // Added serialVersionUID to preserve binary compatibility
    static final long serialVersionUID = -5651165872476709336L;
}

// end of SAXParseException.java
```

47.16 XMLFilter.java

Secuencia 493: org/xml/sax/XMLFilter.java

```java
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// XMLFilter.java - filter SAX2 events.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the Public Domain.
// $Id: XMLFilter.java,v 1.2 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;


/**
 * Interface for an XML filter.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>An XML filter is like an XML reader, except that it obtains its
 * events from another XML reader rather than a primary source like
 * an XML document or database.  Filters can modify a stream of
 * events as they pass on to the final application.</p>
 *
 * <p>The XMLFilterImpl helper class provides a convenient base
 * for creating SAX2 filters, by passing on all {@link org.xml.sax.EntityResolver
 * EntityResolver}, {@link org.xml.sax.DTDHandler DTDHandler},
 * {@link org.xml.sax.ContentHandler ContentHandler} and {@link org.xml.sax.ErrorHandler
 * ErrorHandler} events automatically.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.helpers.XMLFilterImpl
 */
public interface XMLFilter extends XMLReader
{

    /**
     * Set the parent reader.
```

```
     *
     * <p>This method allows the application to link the filter to
     * a parent reader (which may be another filter).  The argument
     * may not be null.</p>
     *
     * @param parent The parent reader.
     */
    public abstract void setParent (XMLReader parent);


    /**
     * Get the parent reader.
     *
     * <p>This method allows the application to query the parent
     * reader (which may be another filter).  It is generally a
     * bad idea to perform any operations on the parent reader
     * directly: they should all pass through this filter.</p>
     *
     * @return The parent filter, or null if none has been set.
     */
    public abstract XMLReader getParent ();

}

// end of XMLFilter.java
```

47.17 XMLReader.java

Secuencia 494: org/xml/sax/XMLReader.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

```

// XMLReader.java - read an XML document.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the Public Domain.
// $Id: XMLReader.java,v 1.3 2004/11/03 22:55:32 jsuttor Exp $

package org.xml.sax;

import java.io.IOException;


/**
 * Interface for reading an XML document using callbacks.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p><strong>Note:</strong> despite its name, this interface does
 * <em>not</em> extend the standard Java {@link java.io.Reader Reader}
 * interface, because reading XML is a fundamentally different activity
 * than reading character data.</p>
 *
 * <p>XMLReader is the interface that an XML parser's SAX2 driver must
 * implement.  This interface allows an application to set and
 * query features and properties in the parser, to register
 * event handlers for document processing, and to initiate
 * a document parse.</p>
 *
 * <p>All SAX interfaces are assumed to be synchronous: the
 * {@link #parse parse} methods must not return until parsing
 * is complete, and readers must wait for an event-handler callback
 * to return before reporting the next event.</p>
 *
 * <p>This interface replaces the (now deprecated) SAX 1.0 {@link
 * org.xml.sax.Parser Parser} interface.  The XMLReader interface
 * contains two important enhancements over the old Parser
 * interface (as well as some minor ones):</p>
 *
 * <ol>
 * <li>it adds a standard way to query and set features and
 *  properties; and</li>
 * <li>it adds Namespace support, which is required for many
 *  higher-level XML standards.</li>
 * </ol>
 *
 * <p>There are adapters available to convert a SAX1 Parser to
 * a SAX2 XMLReader and vice-versa.</p>
 *
```

```
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.XMLFilter
 * @see org.xml.sax.helpers.ParserAdapter
 * @see org.xml.sax.helpers.XMLReaderAdapter
 */
public interface XMLReader
{


    ////////////////////////////////////////////////////////////////////
    // Configuration.
    ////////////////////////////////////////////////////////////////////


    /**
     * Look up the value of a feature flag.
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for an XMLReader to recognize a feature name but
     * temporarily be unable to return its value.
     * Some feature values may be available only in specific
     * contexts, such as before, during, or after a parse.
     * Also, some feature values may not be programmatically accessible.
     * (In the case of an adapter for SAX1 {@link Parser}, there is no
     * implementation-independent way to expose whether the underlying
     * parser is performing validation, expanding external entities,
     * and so forth.) </p>
     *
     * <p>All XMLReaders are required to recognize the
     * http://xml.org/sax/features/namespaces and the
     * http://xml.org/sax/features/namespace-prefixes feature names.</p>
     *
     * <p>Typical usage is something like this:</p>
     *
     * <pre>
     * XMLReader r = new MySAXDriver();
     *
     *                         // try to activate validation
     * try {
     *   r.setFeature("http://xml.org/sax/features/validation", true);
     * } catch (SAXException e) {
     *   System.err.println("Cannot activate validation.");
     * }
     *
     *                         // register event handlers
     * r.setContentHandler(new MyContentHandler());
     * r.setErrorHandler(new MyErrorHandler());
     *
     *                         // parse the first document
     * try {
     *   r.parse("http://www.foo.com/mydoc.xml");
     * } catch (IOException e) {
```

```
     *   System.err.println("I/O exception reading XML document");
     * } catch (SAXException e) {
     *   System.err.println("XML exception reading document.");
     * }
     * </pre>
     *
     * <p>Implementors are free (and encouraged) to invent their own features,
     * using names built on their own URIs.</p>
     *
     * @param name The feature name, which is a fully-qualified URI.
     * @return The current value of the feature (true or false).
     * @exception org.xml.sax.SAXNotRecognizedException If the feature
     *            value can't be assigned or retrieved.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            XMLReader recognizes the feature name but
     *            cannot determine its value at this time.
     * @see #setFeature
     */
    public boolean getFeature (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException;


    /**
     * Set the value of a feature flag.
     *
     * <p>The feature name is any fully-qualified URI.  It is
     * possible for an XMLReader to expose a feature value but
     * to be unable to change the current value.
     * Some feature values may be immutable or mutable only
     * in specific contexts, such as before, during, or after
     * a parse.</p>
     *
     * <p>All XMLReaders are required to support setting
     * http://xml.org/sax/features/namespaces to true and
     * http://xml.org/sax/features/namespace-prefixes to false.</p>
     *
     * @param name The feature name, which is a fully-qualified URI.
     * @param value The requested value of the feature (true or false).
     * @exception org.xml.sax.SAXNotRecognizedException If the feature
     *            value can't be assigned or retrieved.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            XMLReader recognizes the feature name but
     *            cannot set the requested value.
     * @see #getFeature
     */
    public void setFeature (String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException;


    /**
     * Look up the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
```

```
     * possible for an XMLReader to recognize a property name but
     * temporarily be unable to return its value.
     * Some property values may be available only in specific
     * contexts, such as before, during, or after a parse.</p>
     *
     * <p>XMLReaders are not required to recognize any specific
     * property names, though an initial core set is documented for
     * SAX2.</p>
     *
     * <p>Implementors are free (and encouraged) to invent their own properties,
     * using names built on their own URIs.</p>
     *
     * @param name The property name, which is a fully-qualified URI.
     * @return The current value of the property.
     * @exception org.xml.sax.SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            XMLReader recognizes the property name but
     *            cannot determine its value at this time.
     * @see #setProperty
     */
    public Object getProperty (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException;


    /**
     * Set the value of a property.
     *
     * <p>The property name is any fully-qualified URI.  It is
     * possible for an XMLReader to recognize a property name but
     * to be unable to change the current value.
     * Some property values may be immutable or mutable only
     * in specific contexts, such as before, during, or after
     * a parse.</p>
     *
     * <p>XMLReaders are not required to recognize setting
     * any specific property names, though a core set is defined by
     * SAX2.</p>
     *
     * <p>This method is also the standard mechanism for setting
     * extended handlers.</p>
     *
     * @param name The property name, which is a fully-qualified URI.
     * @param value The requested value for the property.
     * @exception org.xml.sax.SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            XMLReader recognizes the property name but
     *            cannot set the requested value.
     */
    public void setProperty (String name, Object value)
        throws SAXNotRecognizedException, SAXNotSupportedException;

```

```java


    ////////////////////////////////////////////////////////////////////
    // Event handlers.
    ////////////////////////////////////////////////////////////////////


    /**
     * Allow an application to register an entity resolver.
     *
     * <p>If the application does not register an entity resolver,
     * the XMLReader will perform its own default resolution.</p>
     *
     * <p>Applications may register a new or different resolver in the
     * middle of a parse, and the SAX parser must begin using the new
     * resolver immediately.</p>
     *
     * @param resolver The entity resolver.
     * @see #getEntityResolver
     */
    public void setEntityResolver (EntityResolver resolver);


    /**
     * Return the current entity resolver.
     *
     * @return The current entity resolver, or null if none
     *         has been registered.
     * @see #setEntityResolver
     */
    public EntityResolver getEntityResolver ();


    /**
     * Allow an application to register a DTD event handler.
     *
     * <p>If the application does not register a DTD handler, all DTD
     * events reported by the SAX parser will be silently ignored.</p>
     *
     * <p>Applications may register a new or different handler in the
     * middle of a parse, and the SAX parser must begin using the new
     * handler immediately.</p>
     *
     * @param handler The DTD handler.
     * @see #getDTDHandler
     */
    public void setDTDHandler (DTDHandler handler);


    /**
     * Return the current DTD handler.
     *
     * @return The current DTD handler, or null if none
```

```
     *         has been registered.
     * @see #setDTDHandler
     */
    public DTDHandler getDTDHandler ();


    /**
     * Allow an application to register a content event handler.
     *
     * <p>If the application does not register a content handler, all
     * content events reported by the SAX parser will be silently
     * ignored.</p>
     *
     * <p>Applications may register a new or different handler in the
     * middle of a parse, and the SAX parser must begin using the new
     * handler immediately.</p>
     *
     * @param handler The content handler.
     * @see #getContentHandler
     */
    public void setContentHandler (ContentHandler handler);


    /**
     * Return the current content handler.
     *
     * @return The current content handler, or null if none
     *         has been registered.
     * @see #setContentHandler
     */
    public ContentHandler getContentHandler ();


    /**
     * Allow an application to register an error event handler.
     *
     * <p>If the application does not register an error handler, all
     * error events reported by the SAX parser will be silently
     * ignored; however, normal processing may not continue.  It is
     * highly recommended that all SAX applications implement an
     * error handler to avoid unexpected bugs.</p>
     *
     * <p>Applications may register a new or different handler in the
     * middle of a parse, and the SAX parser must begin using the new
     * handler immediately.</p>
     *
     * @param handler The error handler.
     * @see #getErrorHandler
     */
    public void setErrorHandler (ErrorHandler handler);


    /**
```

```
     * Return the current error handler.
     *
     * @return The current error handler, or null if none
     *         has been registered.
     * @see #setErrorHandler
     */
    public ErrorHandler getErrorHandler ();



    ////////////////////////////////////////////////////////////////////
    // Parsing.
    ////////////////////////////////////////////////////////////////////

    /**
     * Parse an XML document.
     *
     * <p>The application can use this method to instruct the XML
     * reader to begin parsing an XML document from any valid input
     * source (a character stream, a byte stream, or a URI).</p>
     *
     * <p>Applications may not invoke this method while a parse is in
     * progress (they should create a new XMLReader instead for each
     * nested XML document).  Once a parse is complete, an
     * application may reuse the same XMLReader object, possibly with a
     * different input source.
     * Configuration of the XMLReader object (such as handler bindings and
     * values established for feature flags and properties) is unchanged
     * by completion of a parse, unless the definition of that aspect of
     * the configuration explicitly specifies other behavior.
     * (For example, feature flags or properties exposing
     * characteristics of the document being parsed.)
     * </p>
     *
     * <p>During the parse, the XMLReader will provide information
     * about the XML document through the registered event
     * handlers.</p>
     *
     * <p>This method is synchronous: it will not return until parsing
     * has ended.  If a client application wants to terminate
     * parsing early, it should throw an exception.</p>
     *
     * @param input The input source for the top-level of the
     *        XML document.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     * @see org.xml.sax.InputSource
     * @see #parse(java.lang.String)
     * @see #setEntityResolver
     * @see #setDTDHandler
```

```
     * @see #setContentHandler
     * @see #setErrorHandler
     */
    public void parse (InputSource input)
        throws IOException, SAXException;


    /**
     * Parse an XML document from a system identifier (URI).
     *
     * <p>This method is a shortcut for the common case of reading a
     * document from a system identifier.  It is the exact
     * equivalent of the following:</p>
     *
     * <pre>
     * parse(new InputSource(systemId));
     * </pre>
     *
     * <p>If the system identifier is a URL, it must be fully resolved
     * by the application before it is passed to the parser.</p>
     *
     * @param systemId The system identifier (URI).
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     * @see #parse(org.xml.sax.InputSource)
     */
    public void parse (String systemId)
        throws IOException, SAXException;

}
```

48 Sax Ext (org.xml.sax.ext package)

48.1 Attributes2.java

Secuencia 495: org/xml/sax/ext/Attributes2.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// Attributes2.java - extended Attributes
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: Attributes2.java,v 1.2 2004/11/03 22:49:07 jsuttor Exp $

package org.xml.sax.ext;

import org.xml.sax.Attributes;


/**
 * SAX2 extension to augment the per-attribute information
 * provided though {@link Attributes}.
 * If an implementation supports this extension, the attributes
 * provided in {@link org.xml.sax.ContentHandler#startElement
 * ContentHandler.startElement() } will implement this interface,
 * and the <em>http://xml.org/sax/features/use-attributes2</em>
 * feature flag will have the value <em>true</em>.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p> XMLReader implementations are not required to support this
 * information, and it is not part of core-only SAX2 distributions.</p>
 *
 * <p>Note that if an attribute was defaulted (<em>!isSpecified()</em>)
 * it will of necessity also have been declared (<em>isDeclared()</em>)
 * in the DTD.
 * Similarly if an attribute's type is anything except CDATA, then it
 * must have been declared.
 * </p>
 *
 * @since SAX 2.0 (extensions 1.1 alpha)
 * @author David Brownell
 */
public interface Attributes2 extends Attributes
{
    /**
     * Returns false unless the attribute was declared in the DTD.
     * This helps distinguish two kinds of attributes that SAX reports
```

```
     * as CDATA:  ones that were declared (and hence are usually valid),
     * and those that were not (and which are never valid).
     *
     * @param index The attribute index (zero-based).
     * @return true if the attribute was declared in the DTD,
     *          false otherwise.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not identify an attribute.
     */
    public boolean isDeclared (int index);

    /**
     * Returns false unless the attribute was declared in the DTD.
     * This helps distinguish two kinds of attributes that SAX reports
     * as CDATA:  ones that were declared (and hence are usually valid),
     * and those that were not (and which are never valid).
     *
     * @param qName The XML qualified (prefixed) name.
     * @return true if the attribute was declared in the DTD,
     *          false otherwise.
     * @exception java.lang.IllegalArgumentException When the
     *            supplied name does not identify an attribute.
     */
    public boolean isDeclared (String qName);

    /**
     * Returns false unless the attribute was declared in the DTD.
     * This helps distinguish two kinds of attributes that SAX reports
     * as CDATA:  ones that were declared (and hence are usually valid),
     * and those that were not (and which are never valid).
     *
     * <p>Remember that since DTDs do not "understand" namespaces, the
     * namespace URI associated with an attribute may not have come from
     * the DTD.  The declaration will have applied to the attribute's
     * <em>qName</em>.
     *
     * @param uri The Namespace URI, or the empty string if
     *        the name has no Namespace URI.
     * @param localName The attribute's local name.
     * @return true if the attribute was declared in the DTD,
     *          false otherwise.
     * @exception java.lang.IllegalArgumentException When the
     *            supplied names do not identify an attribute.
     */
    public boolean isDeclared (String uri, String localName);

    /**
     * Returns true unless the attribute value was provided
     * by DTD defaulting.
     *
     * @param index The attribute index (zero-based).
     * @return true if the value was found in the XML text,
     *          false if the value was provided by DTD defaulting.
```

```
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not identify an attribute.
     */
    public boolean isSpecified (int index);

    /**
     * Returns true unless the attribute value was provided
     * by DTD defaulting.
     *
     * <p>Remember that since DTDs do not "understand" namespaces, the
     * namespace URI associated with an attribute may not have come from
     * the DTD.  The declaration will have applied to the attribute's
     * <em>qName</em>.
     *
     * @param uri The Namespace URI, or the empty string if
     *        the name has no Namespace URI.
     * @param localName The attribute's local name.
     * @return true if the value was found in the XML text,
     *          false if the value was provided by DTD defaulting.
     * @exception java.lang.IllegalArgumentException When the
     *            supplied names do not identify an attribute.
     */
    public boolean isSpecified (String uri, String localName);

    /**
     * Returns true unless the attribute value was provided
     * by DTD defaulting.
     *
     * @param qName The XML qualified (prefixed) name.
     * @return true if the value was found in the XML text,
     *          false if the value was provided by DTD defaulting.
     * @exception java.lang.IllegalArgumentException When the
     *            supplied name does not identify an attribute.
     */
    public boolean isSpecified (String qName);
}
```

48.2 Attributes2Impl.java

Secuencia 496: org/xml/sax/ext/Attributes2Impl.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// Attributes2Impl.java - extended AttributesImpl
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: Attributes2Impl.java,v 1.3 2005/02/24 11:20:18 gg156739 Exp $

package org.xml.sax.ext;

import org.xml.sax.Attributes;
import org.xml.sax.helpers.AttributesImpl;


/**
 * SAX2 extension helper for additional Attributes information,
 * implementing the {@link Attributes2} interface.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p>This is not part of core-only SAX2 distributions.</p>
 *
 * <p>The <em>specified</em> flag for each attribute will always
 * be true, unless it has been set to false in the copy constructor
 * or using {@link #setSpecified}.
 * Similarly, the <em>declared</em> flag for each attribute will
 * always be false, except for defaulted attributes (<em>specified</em>
 * is false), non-CDATA attributes, or when it is set to true using
 * {@link #setDeclared}.
 * If you change an attribute's type by hand, you may need to modify
 * its <em>declared</em> flag to match.
 * </p>
 *
 * @since SAX 2.0 (extensions 1.1 alpha)
 * @author David Brownell
 */
public class Attributes2Impl extends AttributesImpl implements Attributes2
{
    private boolean     declared [];
    private boolean     specified [];

```

```java

    /**
     * Construct a new, empty Attributes2Impl object.
     */
    public Attributes2Impl () {
        specified = null;
        declared = null;
    }


    /**
     * Copy an existing Attributes or Attributes2 object.
     * If the object implements Attributes2, values of the
     * <em>specified</em> and <em>declared</em> flags for each
     * attribute are copied.
     * Otherwise the flag values are defaulted to assume no DTD was used,
     * unless there is evidence to the contrary (such as attributes with
     * type other than CDATA, which must have been <em>declared</em>).
     *
     * <p>This constructor is especially useful inside a
     * {@link org.xml.sax.ContentHandler#startElement startElement} event.</p>
     *
     * @param atts The existing Attributes object.
     */
    public Attributes2Impl (Attributes atts)
    {
        super (atts);
    }


    ////////////////////////////////////////////////////////////////////
    // Implementation of Attributes2
    ////////////////////////////////////////////////////////////////////


    /**
     * Returns the current value of the attribute's "declared" flag.
     */
    // javadoc mostly from interface
    public boolean isDeclared (int index)
    {
        if (index < 0 || index >= getLength ())
            throw new ArrayIndexOutOfBoundsException (
                "No attribute at index: " + index);
        return declared [index];
    }


    /**
     * Returns the current value of the attribute's "declared" flag.
     */
    // javadoc mostly from interface
    public boolean isDeclared (String uri, String localName)
```

```
    {
        int index = getIndex (uri, localName);

        if (index < 0)
            throw new IllegalArgumentException (
                "No such attribute: local=" + localName
                + ", namespace=" + uri);
        return declared [index];
    }


    /**
     * Returns the current value of the attribute's "declared" flag.
     */
    // javadoc mostly from interface
    public boolean isDeclared (String qName)
    {
        int index = getIndex (qName);

        if (index < 0)
            throw new IllegalArgumentException (
                "No such attribute: " + qName);
        return declared [index];
    }


    /**
     * Returns the current value of an attribute's "specified" flag.
     *
     * @param index The attribute index (zero-based).
     * @return current flag value
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not identify an attribute.
     */
    public boolean isSpecified (int index)
    {
        if (index < 0 || index >= getLength ())
            throw new ArrayIndexOutOfBoundsException (
                "No attribute at index: " + index);
        return specified [index];
    }


    /**
     * Returns the current value of an attribute's "specified" flag.
     *
     * @param uri The Namespace URI, or the empty string if
     *        the name has no Namespace URI.
     * @param localName The attribute's local name.
     * @return current flag value
     * @exception java.lang.IllegalArgumentException When the
     *            supplied names do not identify an attribute.
     */
```

```
    public boolean isSpecified (String uri, String localName)
    {
        int index = getIndex (uri, localName);

        if (index < 0)
            throw new IllegalArgumentException (
                "No such attribute: local=" + localName
                + ", namespace=" + uri);
        return specified [index];
    }


    /**
     * Returns the current value of an attribute's "specified" flag.
     *
     * @param qName The XML qualified (prefixed) name.
     * @return current flag value
     * @exception java.lang.IllegalArgumentException When the
     *            supplied name does not identify an attribute.
     */
    public boolean isSpecified (String qName)
    {
        int index = getIndex (qName);

        if (index < 0)
            throw new IllegalArgumentException (
                "No such attribute: " + qName);
        return specified [index];
    }


    ////////////////////////////////////////////////////////////////////
    // Manipulators
    ////////////////////////////////////////////////////////////////////


    /**
     * Copy an entire Attributes object.  The "specified" flags are
     * assigned as true, and "declared" flags as false (except when
     * an attribute's type is not CDATA),
     * unless the object is an Attributes2 object.
     * In that case those flag values are all copied.
     *
     * @see AttributesImpl#setAttributes
     */
    public void setAttributes (Attributes atts)
    {
        int length = atts.getLength ();

        super.setAttributes (atts);
        declared = new boolean [length];
        specified = new boolean [length];

```

```
        if (atts instanceof Attributes2) {
            Attributes2 a2 = (Attributes2) atts;
            for (int i = 0; i < length; i++) {
                declared [i] = a2.isDeclared (i);
                specified [i] = a2.isSpecified (i);
            }
        } else {
            for (int i = 0; i < length; i++) {
                declared [i] = !"CDATA".equals (atts.getType (i));
                specified [i] = true;
            }
        }
    }


    /**
     * Add an attribute to the end of the list, setting its
     * "specified" flag to true.  To set that flag's value
     * to false, use {@link #setSpecified}.
     *
     * <p>Unless the attribute <em>type</em> is CDATA, this attribute
     * is marked as being declared in the DTD.  To set that flag's value
     * to true for CDATA attributes, use {@link #setDeclared}.
     *
     * @see AttributesImpl#addAttribute
     */
    public void addAttribute (String uri, String localName, String qName,
                              String type, String value)
    {
        super.addAttribute (uri, localName, qName, type, value);


        int length = getLength ();
        if(specified==null)
        {
            specified = new boolean[length];
            declared = new boolean[length];
        } else if (length > specified.length) {
            boolean     newFlags [];

            newFlags = new boolean [length];
            System.arraycopy (declared, 0, newFlags, 0, declared.length);
            declared = newFlags;

            newFlags = new boolean [length];
            System.arraycopy (specified, 0, newFlags, 0, specified.length);
            specified = newFlags;
        }

        specified [length - 1] = true;
        declared [length - 1] = !"CDATA".equals (type);
    }

```

```java

    // javadoc entirely from superclass
    public void removeAttribute (int index)
    {
        int origMax = getLength () - 1;

        super.removeAttribute (index);
        if (index != origMax) {
            System.arraycopy (declared, index + 1, declared, index,
                    origMax - index);
            System.arraycopy (specified, index + 1, specified, index,
                    origMax - index);
        }
    }


    /**
     * Assign a value to the "declared" flag of a specific attribute.
     * This is normally needed only for attributes of type CDATA,
     * including attributes whose type is changed to or from CDATA.
     *
     * @param index The index of the attribute (zero-based).
     * @param value The desired flag value.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not identify an attribute.
     * @see #setType
     */
    public void setDeclared (int index, boolean value)
    {
        if (index < 0 || index >= getLength ())
            throw new ArrayIndexOutOfBoundsException (
                "No attribute at index: " + index);
        declared [index] = value;
    }


    /**
     * Assign a value to the "specified" flag of a specific attribute.
     * This is the only way this flag can be cleared, except clearing
     * by initialization with the copy constructor.
     *
     * @param index The index of the attribute (zero-based).
     * @param value The desired flag value.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not identify an attribute.
     */
    public void setSpecified (int index, boolean value)
    {
        if (index < 0 || index >= getLength ())
            throw new ArrayIndexOutOfBoundsException (
                "No attribute at index: " + index);
        specified [index] = value;
    }
```

```
}
```

48.3 DeclHandler.java

Secuencia 497: org/xml/sax/ext/DeclHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// DeclHandler.java - Optional handler for DTD declaration events.
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: DeclHandler.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import org.xml.sax.SAXException;


/**
 * SAX2 extension handler for DTD declaration events.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This is an optional extension handler for SAX2 to provide more
 * complete information about DTD declarations in an XML document.
 * XML readers are not required to recognize this handler, and it
```

```
 * is not part of core-only SAX2 distributions.</p>
 *
 * <p>Note that data-related DTD declarations (unparsed entities and
 * notations) are already reported through the {@link
 * org.xml.sax.DTDHandler DTDHandler} interface.</p>
 *
 * <p>If you are using the declaration handler together with a lexical
 * handler, all of the events will occur between the
 * {@link org.xml.sax.ext.LexicalHandler#startDTD startDTD} and the
 * {@link org.xml.sax.ext.LexicalHandler#endDTD endDTD} events.</p>
 *
 * <p>To set the DeclHandler for an XML reader, use the
 * {@link org.xml.sax.XMLReader#setProperty setProperty} method
 * with the property name
 * <code>http://xml.org/sax/properties/declaration-handler</code>
 * and an object implementing this interface (or null) as the value.
 * If the reader does not report declaration events, it will throw a
 * {@link org.xml.sax.SAXNotRecognizedException SAXNotRecognizedException}
 * when you attempt to register the handler.</p>
 *
 * @since SAX 2.0 (extensions 1.0)
 * @author David Megginson
 */
public interface DeclHandler
{

    /**
     * Report an element type declaration.
     *
     * <p>The content model will consist of the string "EMPTY", the
     * string "ANY", or a parenthesised group, optionally followed
     * by an occurrence indicator.  The model will be normalized so
     * that all parameter entities are fully resolved and all whitespace
     * is removed,and will include the enclosing parentheses.  Other
     * normalization (such as removing redundant parentheses or
     * simplifying occurrence indicators) is at the discretion of the
     * parser.</p>
     *
     * @param name The element type name.
     * @param model The content model as a normalized string.
     * @exception SAXException The application may raise an exception.
     */
    public abstract void elementDecl (String name, String model)
        throws SAXException;


    /**
     * Report an attribute type declaration.
     *
     * <p>Only the effective (first) declaration for an attribute will
     * be reported.  The type will be one of the strings "CDATA",
     * "ID", "IDREF", "IDREFS", "NMTOKEN", "NMTOKENS", "ENTITY",
     * "ENTITIES", a parenthesized token group with
```

```
     * the separator "|" and all whitespace removed, or the word
     * "NOTATION" followed by a space followed by a parenthesized
     * token group with all whitespace removed.</p>
     *
     * <p>The value will be the value as reported to applications,
     * appropriately normalized and with entity and character
     * references expanded.  </p>
     *
     * @param eName The name of the associated element.
     * @param aName The name of the attribute.
     * @param type A string representing the attribute type.
     * @param mode A string representing the attribute defaulting mode
     *        ("#IMPLIED", "#REQUIRED", or "#FIXED") or null if
     *        none of these applies.
     * @param value A string representing the attribute's default value,
     *        or null if there is none.
     * @exception SAXException The application may raise an exception.
     */
    public abstract void attributeDecl (String eName,
                                        String aName,
                                        String type,
                                        String mode,
                                        String value)
        throws SAXException;


    /**
     * Report an internal entity declaration.
     *
     * <p>Only the effective (first) declaration for each entity
     * will be reported.  All parameter entities in the value
     * will be expanded, but general entities will not.</p>
     *
     * @param name The name of the entity.  If it is a parameter
     *        entity, the name will begin with '%'.
     * @param value The replacement text of the entity.
     * @exception SAXException The application may raise an exception.
     * @see #externalEntityDecl
     * @see org.xml.sax.DTDHandler#unparsedEntityDecl
     */
    public abstract void internalEntityDecl (String name, String value)
        throws SAXException;


    /**
     * Report a parsed external entity declaration.
     *
     * <p>Only the effective (first) declaration for each entity
     * will be reported.</p>
     *
     * <p>If the system identifier is a URL, the parser must resolve it
     * fully before passing it to the application.</p>
     *
```

```
     * @param name The name of the entity.  If it is a parameter
     *        entity, the name will begin with '%'.
     * @param publicId The entity's public identifier, or null if none
     *        was given.
     * @param systemId The entity's system identifier.
     * @exception SAXException The application may raise an exception.
     * @see #internalEntityDecl
     * @see org.xml.sax.DTDHandler#unparsedEntityDecl
     */
    public abstract void externalEntityDecl (String name, String publicId,
                                             String systemId)
        throws SAXException;

}

// end of DeclHandler.java
```

48.4 DefaultHandler2.java

Secuencia 498: org/xml/sax/ext/DefaultHandler2.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// DefaultHandler2.java - extended DefaultHandler
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: DefaultHandler2.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import java.io.IOException;
```

```
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


/**
 * This class extends the SAX2 base handler class to support the
 * SAX2 {@link LexicalHandler}, {@link DeclHandler}, and
 * {@link EntityResolver2} extensions.  Except for overriding the
 * original SAX1 {@link DefaultHandler#resolveEntity resolveEntity()}
 * method the added handler methods just return.  Subclassers may
 * override everything on a method-by-method basis.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p> <em>Note:</em> this class might yet learn that the
 * <em>ContentHandler.setDocumentLocator()</em> call might be passed a
 * {@link Locator2} object, and that the
 * <em>ContentHandler.startElement()</em> call might be passed a
 * {@link Attributes2} object.
 *
 * @since SAX 2.0 (extensions 1.1 alpha)
 * @author David Brownell
 */
public class DefaultHandler2 extends DefaultHandler
    implements LexicalHandler, DeclHandler, EntityResolver2
{
    /** Constructs a handler which ignores all parsing events. */
    public DefaultHandler2 () { }


    // SAX2 ext-1.0 LexicalHandler

    public void startCDATA ()
    throws SAXException
        {}

    public void endCDATA ()
    throws SAXException
        {}

    public void startDTD (String name, String publicId, String systemId)
    throws SAXException
        {}

    public void endDTD ()
    throws SAXException
        {}

    public void startEntity (String name)
```

```
    throws SAXException
        {}

    public void endEntity (String name)
    throws SAXException
        {}

    public void comment (char ch [], int start, int length)
    throws SAXException
        { }


    // SAX2 ext-1.0 DeclHandler

    public void attributeDecl (String eName, String aName,
            String type, String mode, String value)
    throws SAXException
        {}

    public void elementDecl (String name, String model)
    throws SAXException
        {}

    public void externalEntityDecl (String name,
        String publicId, String systemId)
    throws SAXException
        {}

    public void internalEntityDecl (String name, String value)
    throws SAXException
        {}

    // SAX2 ext-1.1 EntityResolver2

    /**
     * Tells the parser that if no external subset has been declared
     * in the document text, none should be used.
     */
    public InputSource getExternalSubset (String name, String baseURI)
    throws SAXException, IOException
        { return null; }

    /**
     * Tells the parser to resolve the systemId against the baseURI
     * and read the entity text from that resulting absolute URI.
     * Note that because the older
     * {@link DefaultHandler#resolveEntity DefaultHandler.resolveEntity()},
     * method is overridden to call this one, this method may sometimes
     * be invoked with null <em>name</em> and <em>baseURI</em>, and
     * with the <em>systemId</em> already absolutized.
     */
    public InputSource resolveEntity (String name, String publicId,
            String baseURI, String systemId)
```

```
    throws SAXException, IOException
        { return null; }

    // SAX1 EntityResolver

    /**
     * Invokes
     * {@link EntityResolver2#resolveEntity EntityResolver2.resolveEntity()}
     * with null entity name and base URI.
     * You only need to override that method to use this class.
     */
    public InputSource resolveEntity (String publicId, String systemId)
    throws SAXException, IOException
        { return resolveEntity (null, publicId, null, systemId); }
}
```

48.5 EntityResolver2.java

Secuencia 499: org/xml/sax/ext/EntityResolver2.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// EntityResolver2.java - Extended SAX entity resolver.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: EntityResolver2.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import java.io.IOException;

```

```java
import org.xml.sax.EntityResolver;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import org.xml.sax.SAXException;


/**
 * Extended interface for mapping external entity references to input
 * sources, or providing a missing external subset.  The
 * {@link XMLReader#setEntityResolver XMLReader.setEntityResolver()} method
 * is used to provide implementations of this interface to parsers.
 * When a parser uses the methods in this interface, the
 * {@link EntityResolver2#resolveEntity EntityResolver2.resolveEntity()}
 * method (in this interface) is used <em>instead of</em> the older (SAX 1.0)
 * {@link EntityResolver#resolveEntity EntityResolver.resolveEntity()} method.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p>If a SAX application requires the customized handling which this
 * interface defines for external entities, it must ensure that it uses
 * an XMLReader with the
 * <em>http://xml.org/sax/features/use-entity-resolver2</em> feature flag
 * set to <em>true</em> (which is its default value when the feature is
 * recognized).  If that flag is unrecognized, or its value is false,
 * or the resolver does not implement this interface, then only the
 * {@link EntityResolver} method will be used.
 * </p>
 *
 * <p>That supports three categories of application that modify entity
 * resolution.  <em>Old Style</em> applications won't know about this interface;
 * they will provide an EntityResolver.
 * <em>Transitional Mode</em> provide an EntityResolver2 and automatically
 * get the benefit of its methods in any systems (parsers or other tools)
 * supporting it, due to polymorphism.
 * Both <em>Old Style</em> and <em>Transitional Mode</em> applications will
 * work with any SAX2 parser.
 * <em>New style</em> applications will fail to run except on SAX2 parsers
 * that support this particular feature.
 * They will insist that feature flag have a value of "true", and the
 * EntityResolver2 implementation they provide  might throw an exception
 * if the original SAX 1.0 style entity resolution method is invoked.
 * </p>
 *
 * @see org.xml.sax.XMLReader#setEntityResolver
 *
 * @since SAX 2.0 (extensions 1.1 alpha)
 * @author David Brownell
 */
public interface EntityResolver2 extends EntityResolver
{
```

```
    /**
     * Allows applications to provide an external subset for documents
     * that don't explicitly define one.  Documents with DOCTYPE declarations
     * that omit an external subset can thus augment the declarations
     * available for validation, entity processing, and attribute processing
     * (normalization, defaulting, and reporting types including ID).
     * This augmentation is reported
     * through the {@link LexicalHandler#startDTD startDTD()} method as if
     * the document text had originally included the external subset;
     * this callback is made before any internal subset data or errors
     * are reported.</p>
     *
     * <p>This method can also be used with documents that have no DOCTYPE
     * declaration.  When the root element is encountered,
     * but no DOCTYPE declaration has been seen, this method is
     * invoked.  If it returns a value for the external subset, that root
     * element is declared to be the root element, giving the effect of
     * splicing a DOCTYPE declaration at the end the prolog of a document
     * that could not otherwise be valid.  The sequence of parser callbacks
     * in that case logically resembles this:</p>
     *
     * <pre>
     * ... comments and PIs from the prolog (as usual)
     * startDTD ("rootName", source.getPublicId (), source.getSystemId ());
     * startEntity ("[dtd]");
     * ... declarations, comments, and PIs from the external subset
     * endEntity ("[dtd]");
     * endDTD ();
     * ... then the rest of the document (as usual)
     * startElement (..., "rootName", ...);
     * </pre>
     *
     * <p>Note that the InputSource gets no further resolution.
     * Implementations of this method may wish to invoke
     * {@link #resolveEntity resolveEntity()} to gain benefits such as use
     * of local caches of DTD entities.  Also, this method will never be
     * used by a (non-validating) processor that is not including external
     * parameter entities. </p>
     *
     * <p>Uses for this method include facilitating data validation when
     * interoperating with XML processors that would always require
     * undesirable network accesses for external entities, or which for
     * other reasons adopt a "no DTDs" policy.
     * Non-validation motives include forcing documents to include DTDs so
     * that attributes are handled consistently.
     * For example, an XPath processor needs to know which attibutes have
     * type "ID" before it can process a widely used type of reference.</p>
     *
     * <p><strong>Warning:</strong> Returning an external subset modifies
     * the input document.  By providing definitions for general entities,
     * it can make a malformed document appear to be well formed.
     * </p>
     *
```

```
     * @param name Identifies the document root element.  This name comes
     *  from a DOCTYPE declaration (where available) or from the actual
     *  root element.
     * @param baseURI The document's base URI, serving as an additional
     *  hint for selecting the external subset.  This is always an absolute
     *  URI, unless it is null because the XMLReader was given an InputSource
     *  without one.
     *
     * @return An InputSource object describing the new external subset
     *  to be used by the parser, or null to indicate that no external
     *  subset is provided.
     *
     * @exception SAXException Any SAX exception, possibly wrapping
     *  another exception.
     * @exception IOException Probably indicating a failure to create
     *  a new InputStream or Reader, or an illegal URL.
     */
    public InputSource getExternalSubset (String name, String baseURI)
    throws SAXException, IOException;

    /**
     * Allows applications to map references to external entities into input
     * sources, or tell the parser it should use conventional URI resolution.
     * This method is only called for external entities which have been
     * properly declared.
     * This method provides more flexibility than the {@link EntityResolver}
     * interface, supporting implementations of more complex catalogue
     * schemes such as the one defined by the <a href=
        "http://www.oasis-open.org/committees/entity/spec-2001-08-06.html"
        >OASIS XML Catalogs</a> specification.</p>
     *
     * <p>Parsers configured to use this resolver method will call it
     * to determine the input source to use for any external entity
     * being included because of a reference in the XML text.
     * That excludes the document entity, and any external entity returned
     * by {@link #getExternalSubset getExternalSubset()}.
     * When a (non-validating) processor is configured not to include
     * a class of entities (parameter or general) through use of feature
     * flags, this method is not invoked for such entities.  </p>
     *
     * <p>Note that the entity naming scheme used here is the same one
     * used in the {@link LexicalHandler}, or in the {@link
        org.xml.sax.ContentHandler#skippedEntity
        ContentHandler.skippedEntity()}
     * method. </p>
     *
     * @param name Identifies the external entity being resolved.
     *  Either "[dtd]" for the external subset, or a name starting
     *  with "%" to indicate a parameter entity, or else the name of
     *  a general entity.  This is never null when invoked by a SAX2
     *  parser.
     * @param publicId The public identifier of the external entity being
     *  referenced (normalized as required by the XML specification), or
```

```
     *  null if none was supplied.
     * @param baseURI The URI with respect to which relative systemIDs
     *  are interpreted.  This is always an absolute URI, unless it is
     *  null (likely because the XMLReader was given an InputSource without
     *  one).  This URI is defined by the XML specification to be the one
     *  associated with the "&lt;" starting the relevant declaration.
     * @param systemId The system identifier of the external entity
     *  being referenced; either a relative or absolute URI.
     *  This is never null when invoked by a SAX2 parser; only declared
     *  entities, and any external subset, are resolved by such parsers.
     *
     * @return An InputSource object describing the new input source to
     *  be used by the parser.  Returning null directs the parser to
     *  resolve the system ID against the base URI and open a connection
     *  to resulting URI.
     *
     * @exception SAXException Any SAX exception, possibly wrapping
     *  another exception.
     * @exception IOException Probably indicating a failure to create
     *  a new InputStream or Reader, or an illegal URL.
     */
    public InputSource resolveEntity (
            String name,
            String publicId,
            String baseURI,
            String systemId
    ) throws SAXException, IOException;
}
```

48.6 LexicalHandler.java

Secuencia 500: org/xml/sax/ext/LexicalHandler.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 */

// LexicalHandler.java - optional handler for lexical parse events.
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: LexicalHandler.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import org.xml.sax.SAXException;

/**
 * SAX2 extension handler for lexical events.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This is an optional extension handler for SAX2 to provide
 * lexical information about an XML document, such as comments
 * and CDATA section boundaries.
 * XML readers are not required to recognize this handler, and it
 * is not part of core-only SAX2 distributions.</p>
 *
 * <p>The events in the lexical handler apply to the entire document,
 * not just to the document element, and all lexical handler events
 * must appear between the content handler's startDocument and
 * endDocument events.</p>
 *
 * <p>To set the LexicalHandler for an XML reader, use the
 * {@link org.xml.sax.XMLReader#setProperty setProperty} method
 * with the property name
 * <code>http://xml.org/sax/properties/lexical-handler</code>
 * and an object implementing this interface (or null) as the value.
 * If the reader does not report lexical events, it will throw a
 * {@link org.xml.sax.SAXNotRecognizedException SAXNotRecognizedException}
 * when you attempt to register the handler.</p>
 *
 * @since SAX 2.0 (extensions 1.0)
 * @author David Megginson
 */
public interface LexicalHandler
{

    /**
     * Report the start of DTD declarations, if any.
     *
     * <p>This method is intended to report the beginning of the
```

```
     * DOCTYPE declaration; if the document has no DOCTYPE declaration,
     * this method will not be invoked.</p>
     *
     * <p>All declarations reported through
     * {@link org.xml.sax.DTDHandler DTDHandler} or
     * {@link org.xml.sax.ext.DeclHandler DeclHandler} events must appear
     * between the startDTD and {@link #endDTD endDTD} events.
     * Declarations are assumed to belong to the internal DTD subset
     * unless they appear between {@link #startEntity startEntity}
     * and {@link #endEntity endEntity} events.  Comments and
     * processing instructions from the DTD should also be reported
     * between the startDTD and endDTD events, in their original
     * order of (logical) occurrence; they are not required to
     * appear in their correct locations relative to DTDHandler
     * or DeclHandler events, however.</p>
     *
     * <p>Note that the start/endDTD events will appear within
     * the start/endDocument events from ContentHandler and
     * before the first
     * {@link org.xml.sax.ContentHandler#startElement startElement}
     * event.</p>
     *
     * @param name The document type name.
     * @param publicId The declared public identifier for the
     *        external DTD subset, or null if none was declared.
     * @param systemId The declared system identifier for the
     *        external DTD subset, or null if none was declared.
     *        (Note that this is not resolved against the document
     *        base URI.)
     * @exception SAXException The application may raise an
     *            exception.
     * @see #endDTD
     * @see #startEntity
     */
    public abstract void startDTD (String name, String publicId,
                                   String systemId)
        throws SAXException;


    /**
     * Report the end of DTD declarations.
     *
     * <p>This method is intended to report the end of the
     * DOCTYPE declaration; if the document has no DOCTYPE declaration,
     * this method will not be invoked.</p>
     *
     * @exception SAXException The application may raise an exception.
     * @see #startDTD
     */
    public abstract void endDTD ()
        throws SAXException;


```

```
    /**
     * Report the beginning of some internal and external XML entities.
     *
     * <p>The reporting of parameter entities (including
     * the external DTD subset) is optional, and SAX2 drivers that
     * report LexicalHandler events may not implement it; you can use the
     * <code
     * >http://xml.org/sax/features/lexical-handler/parameter-entities</code>
     * feature to query or control the reporting of parameter entities.</p>
     *
     * <p>General entities are reported with their regular names,
     * parameter entities have '%' prepended to their names, and
     * the external DTD subset has the pseudo-entity name "[dtd]".</p>
     *
     * <p>When a SAX2 driver is providing these events, all other
     * events must be properly nested within start/end entity
     * events.  There is no additional requirement that events from
     * {@link org.xml.sax.ext.DeclHandler DeclHandler} or
     * {@link org.xml.sax.DTDHandler DTDHandler} be properly ordered.</p>
     *
     * <p>Note that skipped entities will be reported through the
     * {@link org.xml.sax.ContentHandler#skippedEntity skippedEntity}
     * event, which is part of the ContentHandler interface.</p>
     *
     * <p>Because of the streaming event model that SAX uses, some
     * entity boundaries cannot be reported under any
     * circumstances:</p>
     *
     * <ul>
     * <li>general entities within attribute values</li>
     * <li>parameter entities within declarations</li>
     * </ul>
     *
     * <p>These will be silently expanded, with no indication of where
     * the original entity boundaries were.</p>
     *
     * <p>Note also that the boundaries of character references (which
     * are not really entities anyway) are not reported.</p>
     *
     * <p>All start/endEntity events must be properly nested.
     *
     * @param name The name of the entity.  If it is a parameter
     *        entity, the name will begin with '%', and if it is the
     *        external DTD subset, it will be "[dtd]".
     * @exception SAXException The application may raise an exception.
     * @see #endEntity
     * @see org.xml.sax.ext.DeclHandler#internalEntityDecl
     * @see org.xml.sax.ext.DeclHandler#externalEntityDecl
     */
    public abstract void startEntity (String name)
        throws SAXException;
```

```
    /**
     * Report the end of an entity.
     *
     * @param name The name of the entity that is ending.
     * @exception SAXException The application may raise an exception.
     * @see #startEntity
     */
    public abstract void endEntity (String name)
        throws SAXException;


    /**
     * Report the start of a CDATA section.
     *
     * <p>The contents of the CDATA section will be reported through
     * the regular {@link org.xml.sax.ContentHandler#characters
     * characters} event; this event is intended only to report
     * the boundary.</p>
     *
     * @exception SAXException The application may raise an exception.
     * @see #endCDATA
     */
    public abstract void startCDATA ()
        throws SAXException;


    /**
     * Report the end of a CDATA section.
     *
     * @exception SAXException The application may raise an exception.
     * @see #startCDATA
     */
    public abstract void endCDATA ()
        throws SAXException;


    /**
     * Report an XML comment anywhere in the document.
     *
     * <p>This callback will be used for comments inside or outside the
     * document element, including comments in the external DTD
     * subset (if read).  Comments in the DTD must be properly
     * nested inside start/endDTD and start/endEntity events (if
     * used).</p>
     *
     * @param ch An array holding the characters in the comment.
     * @param start The starting position in the array.
     * @param length The number of characters to use from the array.
     * @exception SAXException The application may raise an exception.
     */
    public abstract void comment (char ch[], int start, int length)
        throws SAXException;

```

```
}

// end of LexicalHandler.java
```

48.7 Locator2.java

Secuencia 501: org/xml/sax/ext/Locator2.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// Locator2.java - extended Locator
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: Locator2.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import org.xml.sax.Locator;


/**
 * SAX2 extension to augment the entity information provided
 * though a {@link Locator}.
 * If an implementation supports this extension, the Locator
 * provided in {@link org.xml.sax.ContentHandler#setDocumentLocator
 * ContentHandler.setDocumentLocator() } will implement this
 * interface, and the
 * <em>http://xml.org/sax/features/use-locator2</em> feature
 * flag will have the value <em>true</em>.
 *
 * <blockquote>
```

```
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p> XMLReader implementations are not required to support this
 * information, and it is not part of core-only SAX2 distributions.</p>
 *
 * @since SAX 2.0 (extensions 1.1 alpha)
 * @author David Brownell
 */
public interface Locator2 extends Locator
{
    /**
     * Returns the version of XML used for the entity.  This will
     * normally be the identifier from the current entity's
     * <em>&lt;?xml version='...' ...?&gt;</em> declaration,
     * or be defaulted by the parser.
     *
     * @return Identifier for the XML version being used to interpret
     * the entity's text, or null if that information is not yet
     * available in the current parsing state.
     */
    public String getXMLVersion ();

    /**
     * Returns the name of the character encoding for the entity.
     * If the encoding was declared externally (for example, in a MIME
     * Content-Type header), that will be the name returned.  Else if there
     * was an <em>&lt;?xml ...encoding='...'?&gt;</em> declaration at
     * the start of the document, that encoding name will be returned.
     * Otherwise the encoding will been inferred (normally to be UTF-8, or
     * some UTF-16 variant), and that inferred name will be returned.
     *
     * <p>When an {@link org.xml.sax.InputSource InputSource} is used
     * to provide an entity's character stream, this method returns the
     * encoding provided in that input stream.
     *
     * <p> Note that some recent W3C specifications require that text
     * in some encodings be normalized, using Unicode Normalization
     * Form C, before processing.  Such normalization must be performed
     * by applications, and would normally be triggered based on the
     * value returned by this method.
     *
     * <p> Encoding names may be those used by the underlying JVM,
     * and comparisons should be case-insensitive.
     *
     * @return Name of the character encoding being used to interpret
     * * the entity's text, or null if this was not provided for a *
     * character stream passed through an InputSource or is otherwise
     * not yet available in the current parsing state.
     */
    public String getEncoding ();
}
```

48.8 Locator2Impl.java

Secuencia 502: org/xml/sax/ext/Locator2Impl.java

```
/*
 * Copyright (c) 2004, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// Locator2Impl.java - extended LocatorImpl
// http://www.saxproject.org
// Public Domain: no warranty.
// $Id: Locator2Impl.java,v 1.2 2004/11/03 22:49:08 jsuttor Exp $

package org.xml.sax.ext;

import org.xml.sax.Locator;
import org.xml.sax.helpers.LocatorImpl;


/**
 * SAX2 extension helper for holding additional Entity information,
 * implementing the {@link Locator2} interface.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * </blockquote>
 *
 * <p> This is not part of core-only SAX2 distributions.</p>
 *
 * @since SAX 2.0.2
 * @author David Brownell
 */
```

```
public class Locator2Impl extends LocatorImpl implements Locator2
{
    private String      encoding;
    private String      version;


    /**
     * Construct a new, empty Locator2Impl object.
     * This will not normally be useful, since the main purpose
     * of this class is to make a snapshot of an existing Locator.
     */
    public Locator2Impl () { }

    /**
     * Copy an existing Locator or Locator2 object.
     * If the object implements Locator2, values of the
     * <em>encoding</em> and <em>version</em>strings are copied,
     * otherwise they set to <em>null</em>.
     *
     * @param locator The existing Locator object.
     */
    public Locator2Impl (Locator locator)
    {
        super (locator);
        if (locator instanceof Locator2) {
            Locator2    l2 = (Locator2) locator;

            version = l2.getXMLVersion ();
            encoding = l2.getEncoding ();
        }
    }

    ////////////////////////////////////////////////////////////////////
    // Locator2 method implementations
    ////////////////////////////////////////////////////////////////////

    /**
     * Returns the current value of the version property.
     *
     * @see #setXMLVersion
     */
    public String getXMLVersion ()
        { return version; }

    /**
     * Returns the current value of the encoding property.
     *
     * @see #setEncoding
     */
    public String getEncoding ()
        { return encoding; }


```

```
    ////////////////////////////////////////////////////////////////
    // Setters
    ////////////////////////////////////////////////////////////////

    /**
     * Assigns the current value of the version property.
     *
     * @param version the new "version" value
     * @see #getXMLVersion
     */
    public void setXMLVersion (String version)
        { this.version = version; }

    /**
     * Assigns the current value of the encoding property.
     *
     * @param encoding the new "encoding" value
     * @see #getEncoding
     */
    public void setEncoding (String encoding)
        { this.encoding = encoding; }
}
```

49 Sax Helpers (org.xml.sax.helpers package)

49.1 AttributeListImpl.java

Secuencia 503: org/xml/sax/helpers/AttributeListImpl.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

```

```
// SAX default implementation for AttributeList.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: AttributeListImpl.java,v 1.2 2004/11/03 22:53:08 jsuttor Exp $

package org.xml.sax.helpers;

import org.xml.sax.AttributeList;

import java.util.Vector;


/**
 * Default implementation for AttributeList.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>AttributeList implements the deprecated SAX1 {@link
 * org.xml.sax.AttributeList AttributeList} interface, and has been
 * replaced by the new SAX2 {@link org.xml.sax.helpers.AttributesImpl
 * AttributesImpl} interface.</p>
 *
 * <p>This class provides a convenience implementation of the SAX
 * {@link org.xml.sax.AttributeList AttributeList} interface.  This
 * implementation is useful both for SAX parser writers, who can use
 * it to provide attributes to the application, and for SAX application
 * writers, who can use it to create a persistent copy of an element's
 * attribute specifications:</p>
 *
 * <pre>
 * private AttributeList myatts;
 *
 * public void startElement (String name, AttributeList atts)
 * {
 *              // create a persistent copy of the attribute list
 *              // for use outside this method
 *   myatts = new AttributeListImpl(atts);
 *   [...]
 * }
 * </pre>
 *
 * <p>Please note that SAX parsers are not required to use this
 * class to provide an implementation of AttributeList; it is
 * supplied only as an optional convenience.  In particular,
 * parser writers are encouraged to invent more efficient
 * implementations.</p>
 *
 * @deprecated This class implements a deprecated interface,
```

```
 *             {@link org.xml.sax.AttributeList AttributeList};
 *             that interface has been replaced by
 *             {@link org.xml.sax.Attributes Attributes},
 *             which is implemented in the
 *             {@link org.xml.sax.helpers.AttributesImpl
 *            AttributesImpl} helper class.
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.AttributeList
 * @see org.xml.sax.DocumentHandler#startElement
 */
public class AttributeListImpl implements AttributeList
{

    /**
     * Create an empty attribute list.
     *
     * <p>This constructor is most useful for parser writers, who
     * will use it to create a single, reusable attribute list that
     * can be reset with the clear method between elements.</p>
     *
     * @see #addAttribute
     * @see #clear
     */
    public AttributeListImpl ()
    {
    }


    /**
     * Construct a persistent copy of an existing attribute list.
     *
     * <p>This constructor is most useful for application writers,
     * who will use it to create a persistent copy of an existing
     * attribute list.</p>
     *
     * @param atts The attribute list to copy
     * @see org.xml.sax.DocumentHandler#startElement
     */
    public AttributeListImpl (AttributeList atts)
    {
        setAttributeList(atts);
    }



    ////////////////////////////////////////////////////////////////////
    // Methods specific to this class.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set the attribute list, discarding previous contents.
```

```
     *
     * <p>This method allows an application writer to reuse an
     * attribute list easily.</p>
     *
     * @param atts The attribute list to copy.
     */
    public void setAttributeList (AttributeList atts)
    {
        int count = atts.getLength();

        clear();

        for (int i = 0; i < count; i++) {
            addAttribute(atts.getName(i), atts.getType(i), atts.getValue(i));
        }
    }


    /**
     * Add an attribute to an attribute list.
     *
     * <p>This method is provided for SAX parser writers, to allow them
     * to build up an attribute list incrementally before delivering
     * it to the application.</p>
     *
     * @param name The attribute name.
     * @param type The attribute type ("NMTOKEN" for an enumeration).
     * @param value The attribute value (must not be null).
     * @see #removeAttribute
     * @see org.xml.sax.DocumentHandler#startElement
     */
    public void addAttribute (String name, String type, String value)
    {
        names.addElement(name);
        types.addElement(type);
        values.addElement(value);
    }


    /**
     * Remove an attribute from the list.
     *
     * <p>SAX application writers can use this method to filter an
     * attribute out of an AttributeList.  Note that invoking this
     * method will change the length of the attribute list and
     * some of the attribute's indices.</p>
     *
     * <p>If the requested attribute is not in the list, this is
     * a no-op.</p>
     *
     * @param name The attribute name.
     * @see #addAttribute
     */
```

```
    public void removeAttribute (String name)
    {
        int i = names.indexOf(name);

        if (i >= 0) {
            names.removeElementAt(i);
            types.removeElementAt(i);
            values.removeElementAt(i);
        }
    }


    /**
     * Clear the attribute list.
     *
     * <p>SAX parser writers can use this method to reset the attribute
     * list between DocumentHandler.startElement events.  Normally,
     * it will make sense to reuse the same AttributeListImpl object
     * rather than allocating a new one each time.</p>
     *
     * @see org.xml.sax.DocumentHandler#startElement
     */
    public void clear ()
    {
        names.removeAllElements();
        types.removeAllElements();
        values.removeAllElements();
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.AttributeList
    ////////////////////////////////////////////////////////////////////


    /**
     * Return the number of attributes in the list.
     *
     * @return The number of attributes in the list.
     * @see org.xml.sax.AttributeList#getLength
     */
    public int getLength ()
    {
        return names.size();
    }


    /**
     * Get the name of an attribute (by position).
     *
     * @param i The position of the attribute in the list.
     * @return The attribute name as a string, or null if there
```

```
     *         is no attribute at that position.
     * @see org.xml.sax.AttributeList#getName(int)
     */
    public String getName (int i)
    {
        if (i < 0) {
            return null;
        }
        try {
            return (String)names.elementAt(i);
        } catch (ArrayIndexOutOfBoundsException e) {
            return null;
        }
    }


    /**
     * Get the type of an attribute (by position).
     *
     * @param i The position of the attribute in the list.
     * @return The attribute type as a string ("NMTOKEN" for an
     *         enumeration, and "CDATA" if no declaration was
     *         read), or null if there is no attribute at
     *         that position.
     * @see org.xml.sax.AttributeList#getType(int)
     */
    public String getType (int i)
    {
        if (i < 0) {
            return null;
        }
        try {
            return (String)types.elementAt(i);
        } catch (ArrayIndexOutOfBoundsException e) {
            return null;
        }
    }


    /**
     * Get the value of an attribute (by position).
     *
     * @param i The position of the attribute in the list.
     * @return The attribute value as a string, or null if
     *         there is no attribute at that position.
     * @see org.xml.sax.AttributeList#getValue(int)
     */
    public String getValue (int i)
    {
        if (i < 0) {
            return null;
        }
        try {
```

```
            return (String)values.elementAt(i);
        } catch (ArrayIndexOutOfBoundsException e) {
            return null;
        }
    }


    /**
     * Get the type of an attribute (by name).
     *
     * @param name The attribute name.
     * @return The attribute type as a string ("NMTOKEN" for an
     *         enumeration, and "CDATA" if no declaration was
     *         read).
     * @see org.xml.sax.AttributeList#getType(java.lang.String)
     */
    public String getType (String name)
    {
        return getType(names.indexOf(name));
    }


    /**
     * Get the value of an attribute (by name).
     *
     * @param name The attribute name.
     * @see org.xml.sax.AttributeList#getValue(java.lang.String)
     */
    public String getValue (String name)
    {
        return getValue(names.indexOf(name));
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    Vector names = new Vector();
    Vector types = new Vector();
    Vector values = new Vector();

}

// end of AttributeListImpl.java
```

49.2 AttributesImpl.java

Secuencia 504: org/xml/sax/helpers/AttributesImpl.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// AttributesImpl.java - default implementation of Attributes.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: AttributesImpl.java,v 1.2 2004/11/03 22:53:08 jsuttor Exp $

package org.xml.sax.helpers;

import org.xml.sax.Attributes;


/**
 * Default implementation of the Attributes interface.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class provides a default implementation of the SAX2
 * {@link org.xml.sax.Attributes Attributes} interface, with the
 * addition of manipulators so that the list can be modified or
 * reused.</p>
 *
 * <p>There are two typical uses of this class:</p>
 *
 * <ol>
 * <li>to take a persistent snapshot of an Attributes object
 *  in a {@link org.xml.sax.ContentHandler#startElement startElement} event; or</li>
```

```
 * <li>to construct or modify an Attributes object in a SAX2 driver or filter.</li>
 * </ol>
 *
 * <p>This class replaces the now-deprecated SAX1 {@link
 * org.xml.sax.helpers.AttributeListImpl AttributeListImpl}
 * class; in addition to supporting the updated Attributes
 * interface rather than the deprecated {@link org.xml.sax.AttributeList
 * AttributeList} interface, it also includes a much more efficient
 * implementation using a single array rather than a set of Vectors.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 */
public class AttributesImpl implements Attributes
{


    ////////////////////////////////////////////////////////////////////
    // Constructors.
    ////////////////////////////////////////////////////////////////////


    /**
     * Construct a new, empty AttributesImpl object.
     */
    public AttributesImpl ()
    {
        length = 0;
        data = null;
    }


    /**
     * Copy an existing Attributes object.
     *
     * <p>This constructor is especially useful inside a
     * {@link org.xml.sax.ContentHandler#startElement startElement} event.</p>
     *
     * @param atts The existing Attributes object.
     */
    public AttributesImpl (Attributes atts)
    {
        setAttributes(atts);
    }




    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.Attributes.
    ////////////////////////////////////////////////////////////////////


    /**
```

```
     * Return the number of attributes in the list.
     *
     * @return The number of attributes in the list.
     * @see org.xml.sax.Attributes#getLength
     */
    public int getLength ()
    {
        return length;
    }


    /**
     * Return an attribute's Namespace URI.
     *
     * @param index The attribute's index (zero-based).
     * @return The Namespace URI, the empty string if none is
     *         available, or null if the index is out of range.
     * @see org.xml.sax.Attributes#getURI
     */
    public String getURI (int index)
    {
        if (index >= 0 && index < length) {
            return data[index*5];
        } else {
            return null;
        }
    }


    /**
     * Return an attribute's local name.
     *
     * @param index The attribute's index (zero-based).
     * @return The attribute's local name, the empty string if
     *         none is available, or null if the index if out of range.
     * @see org.xml.sax.Attributes#getLocalName
     */
    public String getLocalName (int index)
    {
        if (index >= 0 && index < length) {
            return data[index*5+1];
        } else {
            return null;
        }
    }


    /**
     * Return an attribute's qualified (prefixed) name.
     *
     * @param index The attribute's index (zero-based).
     * @return The attribute's qualified name, the empty string if
     *         none is available, or null if the index is out of bounds.
```

```
     * @see org.xml.sax.Attributes#getQName
     */
    public String getQName (int index)
    {
        if (index >= 0 && index < length) {
            return data[index*5+2];
        } else {
            return null;
        }
    }


    /**
     * Return an attribute's type by index.
     *
     * @param index The attribute's index (zero-based).
     * @return The attribute's type, "CDATA" if the type is unknown, or null
     *         if the index is out of bounds.
     * @see org.xml.sax.Attributes#getType(int)
     */
    public String getType (int index)
    {
        if (index >= 0 && index < length) {
            return data[index*5+3];
        } else {
            return null;
        }
    }


    /**
     * Return an attribute's value by index.
     *
     * @param index The attribute's index (zero-based).
     * @return The attribute's value or null if the index is out of bounds.
     * @see org.xml.sax.Attributes#getValue(int)
     */
    public String getValue (int index)
    {
        if (index >= 0 && index < length) {
            return data[index*5+4];
        } else {
            return null;
        }
    }


    /**
     * Look up an attribute's index by Namespace name.
     *
     * <p>In many cases, it will be more efficient to look up the name once and
     * use the index query methods rather than using the name query methods
     * repeatedly.</p>
```

```
     *
     * @param uri The attribute's Namespace URI, or the empty
     *        string if none is available.
     * @param localName The attribute's local name.
     * @return The attribute's index, or -1 if none matches.
     * @see org.xml.sax.Attributes#getIndex(java.lang.String,java.lang.String)
     */
    public int getIndex (String uri, String localName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i].equals(uri) && data[i+1].equals(localName)) {
                return i / 5;
            }
        }
        return -1;
    }


    /**
     * Look up an attribute's index by qualified (prefixed) name.
     *
     * @param qName The qualified name.
     * @return The attribute's index, or -1 if none matches.
     * @see org.xml.sax.Attributes#getIndex(java.lang.String)
     */
    public int getIndex (String qName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i+2].equals(qName)) {
                return i / 5;
            }
        }
        return -1;
    }


    /**
     * Look up an attribute's type by Namespace-qualified name.
     *
     * @param uri The Namespace URI, or the empty string for a name
     *        with no explicit Namespace URI.
     * @param localName The local name.
     * @return The attribute's type, or null if there is no
     *         matching attribute.
     * @see org.xml.sax.Attributes#getType(java.lang.String,java.lang.String)
     */
    public String getType (String uri, String localName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i].equals(uri) && data[i+1].equals(localName)) {
```

```
                return data[i+3];
            }
        }
        return null;
    }


    /**
     * Look up an attribute's type by qualified (prefixed) name.
     *
     * @param qName The qualified name.
     * @return The attribute's type, or null if there is no
     *         matching attribute.
     * @see org.xml.sax.Attributes#getType(java.lang.String)
     */
    public String getType (String qName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i+2].equals(qName)) {
                return data[i+3];
            }
        }
        return null;
    }


    /**
     * Look up an attribute's value by Namespace-qualified name.
     *
     * @param uri The Namespace URI, or the empty string for a name
     *        with no explicit Namespace URI.
     * @param localName The local name.
     * @return The attribute's value, or null if there is no
     *         matching attribute.
     * @see org.xml.sax.Attributes#getValue(java.lang.String,java.lang.String)
     */
    public String getValue (String uri, String localName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i].equals(uri) && data[i+1].equals(localName)) {
                return data[i+4];
            }
        }
        return null;
    }


    /**
     * Look up an attribute's value by qualified (prefixed) name.
     *
     * @param qName The qualified name.
```

```
     * @return The attribute's value, or null if there is no
     *         matching attribute.
     * @see org.xml.sax.Attributes#getValue(java.lang.String)
     */
    public String getValue (String qName)
    {
        int max = length * 5;
        for (int i = 0; i < max; i += 5) {
            if (data[i+2].equals(qName)) {
                return data[i+4];
            }
        }
        return null;
    }



    ////////////////////////////////////////////////////////////////////
    // Manipulators.
    ////////////////////////////////////////////////////////////////////


    /**
     * Clear the attribute list for reuse.
     *
     * <p>Note that little memory is freed by this call:
     * the current array is kept so it can be
     * reused.</p>
     */
    public void clear ()
    {
        if (data != null) {
            for (int i = 0; i < (length * 5); i++)
                data [i] = null;
        }
        length = 0;
    }


    /**
     * Copy an entire Attributes object.
     *
     * <p>It may be more efficient to reuse an existing object
     * rather than constantly allocating new ones.</p>
     *
     * @param atts The attributes to copy.
     */
    public void setAttributes (Attributes atts)
    {
        clear();
        length = atts.getLength();
        if (length > 0) {
            data = new String[length*5];
```

```
            for (int i = 0; i < length; i++) {
                data[i*5] = atts.getURI(i);
                data[i*5+1] = atts.getLocalName(i);
                data[i*5+2] = atts.getQName(i);
                data[i*5+3] = atts.getType(i);
                data[i*5+4] = atts.getValue(i);
            }
        }
    }


    /**
     * Add an attribute to the end of the list.
     *
     * <p>For the sake of speed, this method does no checking
     * to see if the attribute is already in the list: that is
     * the responsibility of the application.</p>
     *
     * @param uri The Namespace URI, or the empty string if
     *        none is available or Namespace processing is not
     *        being performed.
     * @param localName The local name, or the empty string if
     *        Namespace processing is not being performed.
     * @param qName The qualified (prefixed) name, or the empty string
     *        if qualified names are not available.
     * @param type The attribute type as a string.
     * @param value The attribute value.
     */
    public void addAttribute (String uri, String localName, String qName,
                              String type, String value)
    {
        ensureCapacity(length+1);
        data[length*5] = uri;
        data[length*5+1] = localName;
        data[length*5+2] = qName;
        data[length*5+3] = type;
        data[length*5+4] = value;
        length++;
    }


    /**
     * Set an attribute in the list.
     *
     * <p>For the sake of speed, this method does no checking
     * for name conflicts or well-formedness: such checks are the
     * responsibility of the application.</p>
     *
     * @param index The index of the attribute (zero-based).
     * @param uri The Namespace URI, or the empty string if
     *        none is available or Namespace processing is not
     *        being performed.
     * @param localName The local name, or the empty string if
```

```
     *        Namespace processing is not being performed.
     * @param qName The qualified name, or the empty string
     *        if qualified names are not available.
     * @param type The attribute type as a string.
     * @param value The attribute value.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setAttribute (int index, String uri, String localName,
                              String qName, String type, String value)
    {
        if (index >= 0 && index < length) {
            data[index*5] = uri;
            data[index*5+1] = localName;
            data[index*5+2] = qName;
            data[index*5+3] = type;
            data[index*5+4] = value;
        } else {
            badIndex(index);
        }
    }


    /**
     * Remove an attribute from the list.
     *
     * @param index The index of the attribute (zero-based).
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void removeAttribute (int index)
    {
        if (index >= 0 && index < length) {
            if (index < length - 1) {
                System.arraycopy(data, (index+1)*5, data, index*5,
                                 (length-index-1)*5);
            }
            index = (length - 1) * 5;
            data [index++] = null;
            data [index++] = null;
            data [index++] = null;
            data [index++] = null;
            data [index] = null;
            length--;
        } else {
            badIndex(index);
        }
    }


    /**
```

```
     * Set the Namespace URI of a specific attribute.
     *
     * @param index The index of the attribute (zero-based).
     * @param uri The attribute's Namespace URI, or the empty
     *        string for none.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setURI (int index, String uri)
    {
        if (index >= 0 && index < length) {
            data[index*5] = uri;
        } else {
            badIndex(index);
        }
    }


    /**
     * Set the local name of a specific attribute.
     *
     * @param index The index of the attribute (zero-based).
     * @param localName The attribute's local name, or the empty
     *        string for none.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setLocalName (int index, String localName)
    {
        if (index >= 0 && index < length) {
            data[index*5+1] = localName;
        } else {
            badIndex(index);
        }
    }


    /**
     * Set the qualified name of a specific attribute.
     *
     * @param index The index of the attribute (zero-based).
     * @param qName The attribute's qualified name, or the empty
     *        string for none.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setQName (int index, String qName)
    {
        if (index >= 0 && index < length) {
            data[index*5+2] = qName;
```

```
        } else {
            badIndex(index);
        }
    }


    /**
     * Set the type of a specific attribute.
     *
     * @param index The index of the attribute (zero-based).
     * @param type The attribute's type.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setType (int index, String type)
    {
        if (index >= 0 && index < length) {
            data[index*5+3] = type;
        } else {
            badIndex(index);
        }
    }


    /**
     * Set the value of a specific attribute.
     *
     * @param index The index of the attribute (zero-based).
     * @param value The attribute's value.
     * @exception java.lang.ArrayIndexOutOfBoundsException When the
     *            supplied index does not point to an attribute
     *            in the list.
     */
    public void setValue (int index, String value)
    {
        if (index >= 0 && index < length) {
            data[index*5+4] = value;
        } else {
            badIndex(index);
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Internal methods.
    ////////////////////////////////////////////////////////////////////


    /**
     * Ensure the internal array's capacity.
     *
```

```
     * @param n The minimum number of attributes that the array must
     *        be able to hold.
     */
    private void ensureCapacity (int n)    {
        if (n <= 0) {
            return;
        }
        int max;
        if (data == null || data.length == 0) {
            max = 25;
        }
        else if (data.length >= n * 5) {
            return;
        }
        else {
            max = data.length;
        }
        while (max < n * 5) {
            max *= 2;
        }

        String newData[] = new String[max];
        if (length > 0) {
            System.arraycopy(data, 0, newData, 0, length*5);
        }
        data = newData;
    }


    /**
     * Report a bad array index in a manipulator.
     *
     * @param index The index to report.
     * @exception java.lang.ArrayIndexOutOfBoundsException Always.
     */
    private void badIndex (int index)
        throws ArrayIndexOutOfBoundsException
    {
        String msg =
            "Attempt to modify attribute at illegal index: " + index;
        throw new ArrayIndexOutOfBoundsException(msg);
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    int length;
    String data [];

}
```

```

// end of AttributesImpl.java
```

49.3 DefaultHandler.java

Secuencia 505: org/xml/sax/helpers/DefaultHandler.java

```
/*
 * Copyright (c) 2000, 2006, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// DefaultHandler.java - default implementation of the core handlers.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: DefaultHandler.java,v 1.3 2006/04/13 02:06:32 jeffsuttor Exp $

package org.xml.sax.helpers;

import java.io.IOException;

import org.xml.sax.InputSource;
import org.xml.sax.Locator;
import org.xml.sax.Attributes;
import org.xml.sax.EntityResolver;
import org.xml.sax.DTDHandler;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;


/**
```

```
 * Default base class for SAX2 event handlers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class is available as a convenience base class for SAX2
 * applications: it provides default implementations for all of the
 * callbacks in the four core SAX2 handler classes:</p>
 *
 * <ul>
 * <li>{@link org.xml.sax.EntityResolver EntityResolver}</li>
 * <li>{@link org.xml.sax.DTDHandler DTDHandler}</li>
 * <li>{@link org.xml.sax.ContentHandler ContentHandler}</li>
 * <li>{@link org.xml.sax.ErrorHandler ErrorHandler}</li>
 * </ul>
 *
 * <p>Application writers can extend this class when they need to
 * implement only part of an interface; parser writers can
 * instantiate this class to provide default handlers when the
 * application has not supplied its own.</p>
 *
 * <p>This class replaces the deprecated SAX1
 * {@link org.xml.sax.HandlerBase HandlerBase} class.</p>
 *
 * @since SAX 2.0
 * @author David Megginson,
 * @see org.xml.sax.EntityResolver
 * @see org.xml.sax.DTDHandler
 * @see org.xml.sax.ContentHandler
 * @see org.xml.sax.ErrorHandler
 */
public class DefaultHandler
    implements EntityResolver, DTDHandler, ContentHandler, ErrorHandler
{


    ////////////////////////////////////////////////////////////////////
    // Default implementation of the EntityResolver interface.
    ////////////////////////////////////////////////////////////////////

    /**
     * Resolve an external entity.
     *
     * <p>Always return null, so that the parser will use the system
     * identifier provided in the XML document.  This method implements
     * the SAX default behaviour: application writers can override it
     * in a subclass to do special translations such as catalog lookups
     * or URI redirection.</p>
     *
```

```
     * @param publicId The public identifier, or null if none is
     *                 available.
     * @param systemId The system identifier provided in the XML
     *                 document.
     * @return The new input source, or null to require the
     *         default behaviour.
     * @exception java.io.IOException If there is an error setting
     *            up the new input source.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.EntityResolver#resolveEntity
     */
    public InputSource resolveEntity (String publicId, String systemId)
        throws IOException, SAXException
    {
        return null;
    }



    ////////////////////////////////////////////////////////////////////
    // Default implementation of DTDHandler interface.
    ////////////////////////////////////////////////////////////////////


    /**
     * Receive notification of a notation declaration.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass if they wish to keep track of the notations
     * declared in a document.</p>
     *
     * @param name The notation name.
     * @param publicId The notation public identifier, or null if not
     *                 available.
     * @param systemId The notation system identifier.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DTDHandler#notationDecl
     */
    public void notationDecl (String name, String publicId, String systemId)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of an unparsed entity declaration.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to keep track of the unparsed entities
     * declared in a document.</p>
```

```
     *
     * @param name The entity name.
     * @param publicId The entity public identifier, or null if not
     *                 available.
     * @param systemId The entity system identifier.
     * @param notationName The name of the associated notation.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.DTDHandler#unparsedEntityDecl
     */
    public void unparsedEntityDecl (String name, String publicId,
                                    String systemId, String notationName)
        throws SAXException
    {
        // no op
    }



    ////////////////////////////////////////////////////////////////////
    // Default implementation of ContentHandler interface.
    ////////////////////////////////////////////////////////////////////


    /**
     * Receive a Locator object for document events.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass if they wish to store the locator for use
     * with other document events.</p>
     *
     * @param locator A locator for all SAX document events.
     * @see org.xml.sax.ContentHandler#setDocumentLocator
     * @see org.xml.sax.Locator
     */
    public void setDocumentLocator (Locator locator)
    {
        // no op
    }


    /**
     * Receive notification of the beginning of the document.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the beginning
     * of a document (such as allocating the root node of a tree or
     * creating an output file).</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#startDocument
     */
```

```
    public void startDocument ()
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the end of the document.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the end
     * of a document (such as finalising a tree or closing an output
     * file).</p>
     *
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#endDocument
     */
    public void endDocument ()
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the start of a Namespace mapping.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the start of
     * each Namespace prefix scope (such as storing the prefix mapping).</p>
     *
     * @param prefix The Namespace prefix being declared.
     * @param uri The Namespace URI mapped to the prefix.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#startPrefixMapping
     */
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the end of a Namespace mapping.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the end of
     * each prefix mapping.</p>
     *
```

```
     * @param prefix The Namespace prefix being declared.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#endPrefixMapping
     */
    public void endPrefixMapping (String prefix)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the start of an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the start of
     * each element (such as allocating a new tree node or writing
     * output to a file).</p>
     *
     * @param uri The Namespace URI, or the empty string if the
     *        element has no Namespace URI or if Namespace
     *        processing is not being performed.
     * @param localName The local name (without prefix), or the
     *        empty string if Namespace processing is not being
     *        performed.
     * @param qName The qualified name (with prefix), or the
     *        empty string if qualified names are not available.
     * @param attributes The attributes attached to the element.  If
     *        there are no attributes, it shall be an empty
     *        Attributes object.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#startElement
     */
    public void startElement (String uri, String localName,
                              String qName, Attributes attributes)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of the end of an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions at the end of
     * each element (such as finalising a tree node or writing
     * output to a file).</p>
     *
     * @param uri The Namespace URI, or the empty string if the
     *        element has no Namespace URI or if Namespace
```

```
     *        processing is not being performed.
     * @param localName The local name (without prefix), or the
     *        empty string if Namespace processing is not being
     *        performed.
     * @param qName The qualified name (with prefix), or the
     *        empty string if qualified names are not available.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#endElement
     */
    public void endElement (String uri, String localName, String qName)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of character data inside an element.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method to take specific actions for each chunk of character data
     * (such as adding the data to a node or buffer, or printing it to
     * a file).</p>
     *
     * @param ch The characters.
     * @param start The start position in the character array.
     * @param length The number of characters to use from the
     *               character array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#characters
     */
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of ignorable whitespace in element content.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method to take specific actions for each chunk of ignorable
     * whitespace (such as adding data to a node or buffer, or printing
     * it to a file).</p>
     *
     * @param ch The whitespace characters.
     * @param start The start position in the character array.
     * @param length The number of characters to use from the
     *               character array.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
```

```
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#ignorableWhitespace
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of a processing instruction.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions for each
     * processing instruction, such as setting status variables or
     * invoking other methods.</p>
     *
     * @param target The processing instruction target.
     * @param data The processing instruction data, or null if
     *             none is supplied.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#processingInstruction
     */
    public void processingInstruction (String target, String data)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of a skipped entity.
     *
     * <p>By default, do nothing.  Application writers may override this
     * method in a subclass to take specific actions for each
     * processing instruction, such as setting status variables or
     * invoking other methods.</p>
     *
     * @param name The name of the skipped entity.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ContentHandler#processingInstruction
     */
    public void skippedEntity (String name)
        throws SAXException
    {
        // no op
    }
```

```
    ////////////////////////////////////////////////////////////////////
    // Default implementation of the ErrorHandler interface.
    ////////////////////////////////////////////////////////////////////


    /**
     * Receive notification of a parser warning.
     *
     * <p>The default implementation does nothing.  Application writers
     * may override this method in a subclass to take specific actions
     * for each warning, such as inserting the message in a log file or
     * printing it to the console.</p>
     *
     * @param e The warning information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#warning
     * @see org.xml.sax.SAXParseException
     */
    public void warning (SAXParseException e)
        throws SAXException
    {
        // no op
    }


    /**
     * Receive notification of a recoverable parser error.
     *
     * <p>The default implementation does nothing.  Application writers
     * may override this method in a subclass to take specific actions
     * for each error, such as inserting the message in a log file or
     * printing it to the console.</p>
     *
     * @param e The error information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#warning
     * @see org.xml.sax.SAXParseException
     */
    public void error (SAXParseException e)
        throws SAXException
    {
        // no op
    }


    /**
     * Report a fatal XML parsing error.
     *
     * <p>The default implementation throws a SAXParseException.
     * Application writers may override this method in a subclass if
     * they need to take specific actions for each fatal error (such as
```

```
     * collecting all of the errors into a single report): in any case,
     * the application must stop all regular processing when this
     * method is invoked, since the document is no longer reliable, and
     * the parser may no longer report parsing events.</p>
     *
     * @param e The error information encoded as an exception.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @see org.xml.sax.ErrorHandler#fatalError
     * @see org.xml.sax.SAXParseException
     */
    public void fatalError (SAXParseException e)
        throws SAXException
    {
        throw e;
    }

}

// end of DefaultHandler.java
```

49.4 LocatorImpl.java

Secuencia 506: org/xml/sax/helpers/LocatorImpl.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX default implementation for Locator.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: LocatorImpl.java,v 1.2 2004/11/03 22:53:09 jsuttor Exp $
```

```

package org.xml.sax.helpers;

import org.xml.sax.Locator;


/**
 * Provide an optional convenience implementation of Locator.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class is available mainly for application writers, who
 * can use it to make a persistent snapshot of a locator at any
 * point during a document parse:</p>
 *
 * <pre>
 * Locator locator;
 * Locator startloc;
 *
 * public void setLocator (Locator locator)
 * {
 *         // note the locator
 *   this.locator = locator;
 * }
 *
 * public void startDocument ()
 * {
 *         // save the location of the start of the document
 *         // for future use.
 *   Locator startloc = new LocatorImpl(locator);
 * }
 *</pre>
 *
 * <p>Normally, parser writers will not use this class, since it
 * is more efficient to provide location information only when
 * requested, rather than constantly updating a Locator object.</p>
 *
 * @since SAX 1.0
 * @author David Megginson
 * @see org.xml.sax.Locator Locator
 */
public class LocatorImpl implements Locator
{


    /**
     * Zero-argument constructor.
     *
```

```
     * <p>This will not normally be useful, since the main purpose
     * of this class is to make a snapshot of an existing Locator.</p>
     */
    public LocatorImpl ()
    {
    }


    /**
     * Copy constructor.
     *
     * <p>Create a persistent copy of the current state of a locator.
     * When the original locator changes, this copy will still keep
     * the original values (and it can be used outside the scope of
     * DocumentHandler methods).</p>
     *
     * @param locator The locator to copy.
     */
    public LocatorImpl (Locator locator)
    {
        setPublicId(locator.getPublicId());
        setSystemId(locator.getSystemId());
        setLineNumber(locator.getLineNumber());
        setColumnNumber(locator.getColumnNumber());
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.Locator
    ////////////////////////////////////////////////////////////////////


    /**
     * Return the saved public identifier.
     *
     * @return The public identifier as a string, or null if none
     *         is available.
     * @see org.xml.sax.Locator#getPublicId
     * @see #setPublicId
     */
    public String getPublicId ()
    {
        return publicId;
    }


    /**
     * Return the saved system identifier.
     *
     * @return The system identifier as a string, or null if none
     *         is available.
     * @see org.xml.sax.Locator#getSystemId
```

```
     * @see #setSystemId
     */
    public String getSystemId ()
    {
        return systemId;
    }


    /**
     * Return the saved line number (1-based).
     *
     * @return The line number as an integer, or -1 if none is available.
     * @see org.xml.sax.Locator#getLineNumber
     * @see #setLineNumber
     */
    public int getLineNumber ()
    {
        return lineNumber;
    }


    /**
     * Return the saved column number (1-based).
     *
     * @return The column number as an integer, or -1 if none is available.
     * @see org.xml.sax.Locator#getColumnNumber
     * @see #setColumnNumber
     */
    public int getColumnNumber ()
    {
        return columnNumber;
    }



    ////////////////////////////////////////////////////////////////////
    // Setters for the properties (not in org.xml.sax.Locator)
    ////////////////////////////////////////////////////////////////////


    /**
     * Set the public identifier for this locator.
     *
     * @param publicId The new public identifier, or null
     *        if none is available.
     * @see #getPublicId
     */
    public void setPublicId (String publicId)
    {
        this.publicId = publicId;
    }


```

```
    /**
     * Set the system identifier for this locator.
     *
     * @param systemId The new system identifier, or null
     *        if none is available.
     * @see #getSystemId
     */
    public void setSystemId (String systemId)
    {
        this.systemId = systemId;
    }


    /**
     * Set the line number for this locator (1-based).
     *
     * @param lineNumber The line number, or -1 if none is available.
     * @see #getLineNumber
     */
    public void setLineNumber (int lineNumber)
    {
        this.lineNumber = lineNumber;
    }


    /**
     * Set the column number for this locator (1-based).
     *
     * @param columnNumber The column number, or -1 if none is available.
     * @see #getColumnNumber
     */
    public void setColumnNumber (int columnNumber)
    {
        this.columnNumber = columnNumber;
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    private String publicId;
    private String systemId;
    private int lineNumber;
    private int columnNumber;

}

// end of LocatorImpl.java
```

49.5 NamespaceSupport.java

Secuencia 507: org/xml/sax/helpers/NamespaceSupport.java

```
/*
 * Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// NamespaceSupport.java - generic Namespace support for SAX.
// http://www.saxproject.org
// Written by David Megginson
// This class is in the Public Domain.  NO WARRANTY!
// $Id: NamespaceSupport.java,v 1.5 2004/11/03 22:53:09 jsuttor Exp $

package org.xml.sax.helpers;

import java.util.ArrayList;
import java.util.Collections;
import java.util.EmptyStackException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Encapsulate Namespace logic for use by applications using SAX,
 * or internally by SAX drivers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
```

```
 *
 * <p>This class encapsulates the logic of Namespace processing: it
 * tracks the declarations currently in force for each context and
 * automatically processes qualified XML names into their Namespace
 * parts; it can also be used in reverse for generating XML qnames
 * from Namespaces.</p>
 *
 * <p>Namespace support objects are reusable, but the reset method
 * must be invoked between each session.</p>
 *
 * <p>Here is a simple session:</p>
 *
 * <pre>
 * String parts[] = new String[3];
 * NamespaceSupport support = new NamespaceSupport();
 *
 * support.pushContext();
 * support.declarePrefix("", "http://www.w3.org/1999/xhtml");
 * support.declarePrefix("dc", "http://www.purl.org/dc#");
 *
 * parts = support.processName("p", parts, false);
 * System.out.println("Namespace URI: " + parts[0]);
 * System.out.println("Local name: " + parts[1]);
 * System.out.println("Raw name: " + parts[2]);
 *
 * parts = support.processName("dc:title", parts, false);
 * System.out.println("Namespace URI: " + parts[0]);
 * System.out.println("Local name: " + parts[1]);
 * System.out.println("Raw name: " + parts[2]);
 *
 * support.popContext();
 * </pre>
 *
 * <p>Note that this class is optimized for the use case where most
 * elements do not contain Namespace declarations: if the same
 * prefix/URI mapping is repeated for each context (for example), this
 * class will be somewhat less efficient.</p>
 *
 * <p>Although SAX drivers (parsers) may choose to use this class to
 * implement namespace handling, they are not required to do so.
 * Applications must track namespace information themselves if they
 * want to use namespace information.
 *
 * @since SAX 2.0
 * @author David Megginson
 */
public class NamespaceSupport
{


    ////////////////////////////////////////////////////////////////////
    // Constants.
    ////////////////////////////////////////////////////////////////////
```

```java


    /**
     * The XML Namespace URI as a constant.
     * The value is <code>http://www.w3.org/XML/1998/namespace</code>
     * as defined in the "Namespaces in XML" * recommendation.
     *
     * <p>This is the Namespace URI that is automatically mapped
     * to the "xml" prefix.</p>
     */
    public final static String XMLNS =
        "http://www.w3.org/XML/1998/namespace";


    /**
     * The namespace declaration URI as a constant.
     * The value is <code>http://www.w3.org/xmlns/2000/</code>, as defined
     * in a backwards-incompatible erratum to the "Namespaces in XML"
     * recommendation.  Because that erratum postdated SAX2, SAX2 defaults
     * to the original recommendation, and does not normally use this URI.
     *
     *
     * <p>This is the Namespace URI that is optionally applied to
     * <em>xmlns</em> and <em>xmlns:*</em> attributes, which are used to
     * declare namespaces.  </p>
     *
     * @since SAX 2.1alpha
     * @see #setNamespaceDeclUris
     * @see #isNamespaceDeclUris
     */
    public final static String NSDECL =
        "http://www.w3.org/xmlns/2000/";


    /**
     * An empty enumeration.
     */
    private final static Enumeration EMPTY_ENUMERATION =
            Collections.enumeration(new ArrayList<String>());


    ////////////////////////////////////////////////////////////////////
    // Constructor.
    ////////////////////////////////////////////////////////////////////


    /**
     * Create a new Namespace support object.
     */
    public NamespaceSupport ()
    {
        reset();
    }
```

```java



    ////////////////////////////////////////////////////////////////
    // Context management.
    ////////////////////////////////////////////////////////////////


    /**
     * Reset this Namespace support object for reuse.
     *
     * <p>It is necessary to invoke this method before reusing the
     * Namespace support object for a new session.  If namespace
     * declaration URIs are to be supported, that flag must also
     * be set to a non-default value.
     * </p>
     *
     * @see #setNamespaceDeclUris
     */
    public void reset ()
    {
        contexts = new Context[32];
        namespaceDeclUris = false;
        contextPos = 0;
        contexts[contextPos] = currentContext = new Context();
        currentContext.declarePrefix("xml", XMLNS);
    }


    /**
     * Start a new Namespace context.
     * The new context will automatically inherit
     * the declarations of its parent context, but it will also keep
     * track of which declarations were made within this context.
     *
     * <p>Event callback code should start a new context once per element.
     * This means being ready to call this in either of two places.
     * For elements that don't include namespace declarations, the
     * <em>ContentHandler.startElement()</em> callback is the right place.
     * For elements with such a declaration, it'd done in the first
     * <em>ContentHandler.startPrefixMapping()</em> callback.
     * A boolean flag can be used to
     * track whether a context has been started yet.  When either of
     * those methods is called, it checks the flag to see if a new context
     * needs to be started.  If so, it starts the context and sets the
     * flag.  After <em>ContentHandler.startElement()</em>
     * does that, it always clears the flag.
     *
     * <p>Normally, SAX drivers would push a new context at the beginning
     * of each XML element.  Then they perform a first pass over the
     * attributes to process all namespace declarations, making
     * <em>ContentHandler.startPrefixMapping()</em> callbacks.
     * Then a second pass is made, to determine the namespace-qualified
```

```
     * names for all attributes and for the element name.
     * Finally all the information for the
     * <em>ContentHandler.startElement()</em> callback is available,
     * so it can then be made.
     *
     * <p>The Namespace support object always starts with a base context
     * already in force: in this context, only the "xml" prefix is
     * declared.</p>
     *
     * @see org.xml.sax.ContentHandler
     * @see #popContext
     */
    public void pushContext ()
    {
        int max = contexts.length;

        contextPos++;

                                // Extend the array if necessary
        if (contextPos >= max) {
            Context newContexts[] = new Context[max*2];
            System.arraycopy(contexts, 0, newContexts, 0, max);
            max *= 2;
            contexts = newContexts;
        }

                                // Allocate the context if necessary.
        currentContext = contexts[contextPos];
        if (currentContext == null) {
            contexts[contextPos] = currentContext = new Context();
        }

                                // Set the parent, if any.
        if (contextPos > 0) {
            currentContext.setParent(contexts[contextPos - 1]);
        }
    }


    /**
     * Revert to the previous Namespace context.
     *
     * <p>Normally, you should pop the context at the end of each
     * XML element.  After popping the context, all Namespace prefix
     * mappings that were previously in force are restored.</p>
     *
     * <p>You must not attempt to declare additional Namespace
     * prefixes after popping a context, unless you push another
     * context first.</p>
     *
     * @see #pushContext
     */
    public void popContext ()
```

```
    {
        contexts[contextPos].clear();
        contextPos--;
        if (contextPos < 0) {
            throw new EmptyStackException();
        }
        currentContext = contexts[contextPos];
    }



    ////////////////////////////////////////////////////////////////////
    // Operations within a context.
    ////////////////////////////////////////////////////////////////////


    /**
     * Declare a Namespace prefix.  All prefixes must be declared
     * before they are referenced.  For example, a SAX driver (parser)
     * would scan an element's attributes
     * in two passes:  first for namespace declarations,
     * then a second pass using {@link #processName processName()} to
     * interpret prefixes against (potentially redefined) prefixes.
     *
     * <p>This method declares a prefix in the current Namespace
     * context; the prefix will remain in force until this context
     * is popped, unless it is shadowed in a descendant context.</p>
     *
     * <p>To declare the default element Namespace, use the empty string as
     * the prefix.</p>
     *
     * <p>Note that there is an asymmetry in this library: {@link
     * #getPrefix getPrefix} will not return the "" prefix,
     * even if you have declared a default element namespace.
     * To check for a default namespace,
     * you have to look it up explicitly using {@link #getURI getURI}.
     * This asymmetry exists to make it easier to look up prefixes
     * for attribute names, where the default prefix is not allowed.</p>
     *
     * @param prefix The prefix to declare, or the empty string to
     *  indicate the default element namespace.  This may never have
     *  the value "xml" or "xmlns".
     * @param uri The Namespace URI to associate with the prefix.
     * @return true if the prefix was legal, false otherwise
     *
     * @see #processName
     * @see #getURI
     * @see #getPrefix
     */
    public boolean declarePrefix (String prefix, String uri)
    {
        if (prefix.equals("xml") || prefix.equals("xmlns")) {
            return false;
```

```
        } else {
            currentContext.declarePrefix(prefix, uri);
            return true;
        }
    }


    /**
     * Process a raw XML qualified name, after all declarations in the
     * current context have been handled by {@link #declarePrefix
     * declarePrefix()}.
     *
     * <p>This method processes a raw XML qualified name in the
     * current context by removing the prefix and looking it up among
     * the prefixes currently declared.  The return value will be the
     * array supplied by the caller, filled in as follows:</p>
     *
     * <dl>
     * <dt>parts[0]</dt>
     * <dd>The Namespace URI, or an empty string if none is
     *  in use.</dd>
     * <dt>parts[1]</dt>
     * <dd>The local name (without prefix).</dd>
     * <dt>parts[2]</dt>
     * <dd>The original raw name.</dd>
     * </dl>
     *
     * <p>All of the strings in the array will be internalized.  If
     * the raw name has a prefix that has not been declared, then
     * the return value will be null.</p>
     *
     * <p>Note that attribute names are processed differently than
     * element names: an unprefixed element name will receive the
     * default Namespace (if any), while an unprefixed attribute name
     * will not.</p>
     *
     * @param qName The XML qualified name to be processed.
     * @param parts An array supplied by the caller, capable of
     *        holding at least three members.
     * @param isAttribute A flag indicating whether this is an
     *        attribute name (true) or an element name (false).
     * @return The supplied array holding three internalized strings
     *        representing the Namespace URI (or empty string), the
     *        local name, and the XML qualified name; or null if there
     *        is an undeclared prefix.
     * @see #declarePrefix
     * @see java.lang.String#intern */
    public String [] processName (String qName, String parts[],
                                  boolean isAttribute)
    {
        String myParts[] = currentContext.processName(qName, isAttribute);
        if (myParts == null) {
            return null;
```

```
        } else {
            parts[0] = myParts[0];
            parts[1] = myParts[1];
            parts[2] = myParts[2];
            return parts;
        }
    }


    /**
     * Look up a prefix and get the currently-mapped Namespace URI.
     *
     * <p>This method looks up the prefix in the current context.
     * Use the empty string ("") for the default Namespace.</p>
     *
     * @param prefix The prefix to look up.
     * @return The associated Namespace URI, or null if the prefix
     *         is undeclared in this context.
     * @see #getPrefix
     * @see #getPrefixes
     */
    public String getURI (String prefix)
    {
        return currentContext.getURI(prefix);
    }


    /**
     * Return an enumeration of all prefixes whose declarations are
     * active in the current context.
     * This includes declarations from parent contexts that have
     * not been overridden.
     *
     * <p><strong>Note:</strong> if there is a default prefix, it will not be
     * returned in this enumeration; check for the default prefix
     * using the {@link #getURI getURI} with an argument of "".</p>
     *
     * @return An enumeration of prefixes (never empty).
     * @see #getDeclaredPrefixes
     * @see #getURI
     */
    public Enumeration getPrefixes ()
    {
        return currentContext.getPrefixes();
    }


    /**
     * Return one of the prefixes mapped to a Namespace URI.
     *
     * <p>If more than one prefix is currently mapped to the same
     * URI, this method will make an arbitrary selection; if you
     * want all of the prefixes, use the {@link #getPrefixes}
```

```
     * method instead.</p>
     *
     * <p><strong>Note:</strong> this will never return the empty (default) prefix;
     * to check for a default prefix, use the {@link #getURI getURI}
     * method with an argument of "".</p>
     *
     * @param uri the namespace URI
     * @return one of the prefixes currently mapped to the URI supplied,
     *         or null if none is mapped or if the URI is assigned to
     *         the default namespace
     * @see #getPrefixes(java.lang.String)
     * @see #getURI
     */
    public String getPrefix (String uri)
    {
        return currentContext.getPrefix(uri);
    }


    /**
     * Return an enumeration of all prefixes for a given URI whose
     * declarations are active in the current context.
     * This includes declarations from parent contexts that have
     * not been overridden.
     *
     * <p>This method returns prefixes mapped to a specific Namespace
     * URI.  The xml: prefix will be included.  If you want only one
     * prefix that's mapped to the Namespace URI, and you don't care
     * which one you get, use the {@link #getPrefix getPrefix}
     *  method instead.</p>
     *
     * <p><strong>Note:</strong> the empty (default) prefix is <em>never</em> included
     * in this enumeration; to check for the presence of a default
     * Namespace, use the {@link #getURI getURI} method with an
     * argument of "".</p>
     *
     * @param uri The Namespace URI.
     * @return An enumeration of prefixes (never empty).
     * @see #getPrefix
     * @see #getDeclaredPrefixes
     * @see #getURI
     */
    public Enumeration getPrefixes (String uri)
    {
        List<String> prefixes = new ArrayList<>();
        Enumeration allPrefixes = getPrefixes();
        while (allPrefixes.hasMoreElements()) {
            String prefix = (String)allPrefixes.nextElement();
            if (uri.equals(getURI(prefix))) {
                prefixes.add(prefix);
            }
        }
        return Collections.enumeration(prefixes);
```

```
    }


    /**
     * Return an enumeration of all prefixes declared in this context.
     *
     * <p>The empty (default) prefix will be included in this
     * enumeration; note that this behaviour differs from that of
     * {@link #getPrefix} and {@link #getPrefixes}.</p>
     *
     * @return An enumeration of all prefixes declared in this
     *         context.
     * @see #getPrefixes
     * @see #getURI
     */
    public Enumeration getDeclaredPrefixes ()
    {
        return currentContext.getDeclaredPrefixes();
    }

    /**
     * Controls whether namespace declaration attributes are placed
     * into the {@link #NSDECL NSDECL} namespace
     * by {@link #processName processName()}.  This may only be
     * changed before any contexts have been pushed.
     *
     * @since SAX 2.1alpha
     *
     * @exception IllegalStateException when attempting to set this
     *  after any context has been pushed.
     */
    public void setNamespaceDeclUris (boolean value)
    {
        if (contextPos != 0)
            throw new IllegalStateException ();
        if (value == namespaceDeclUris)
            return;
        namespaceDeclUris = value;
        if (value)
            currentContext.declarePrefix ("xmlns", NSDECL);
        else {
            contexts[contextPos] = currentContext = new Context();
            currentContext.declarePrefix("xml", XMLNS);
        }
    }

    /**
     * Returns true if namespace declaration attributes are placed into
     * a namespace.  This behavior is not the default.
     *
     * @since SAX 2.1alpha
     */
    public boolean isNamespaceDeclUris ()
```

```
        { return namespaceDeclUris; }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    private Context contexts[];
    private Context currentContext;
    private int contextPos;
    private boolean namespaceDeclUris;


    ////////////////////////////////////////////////////////////////////
    // Internal classes.
    ////////////////////////////////////////////////////////////////////

    /**
     * Internal class for a single Namespace context.
     *
     * <p>This module caches and reuses Namespace contexts,
     * so the number allocated
     * will be equal to the element depth of the document, not to the total
     * number of elements (i.e. 5-10 rather than tens of thousands).
     * Also, data structures used to represent contexts are shared when
     * possible (child contexts without declarations) to further reduce
     * the amount of memory that's consumed.
     * </p>
     */
    final class Context {

        /**
         * Create the root-level Namespace context.
         */
        Context ()
        {
            copyTables();
        }


        /**
         * (Re)set the parent of this Namespace context.
         * The context must either have been freshly constructed,
         * or must have been cleared.
         *
         * @param context The parent Namespace context object.
         */
        void setParent (Context parent)
        {
            this.parent = parent;
            declarations = null;
            prefixTable = parent.prefixTable;
```

```
            uriTable = parent.uriTable;
            elementNameTable = parent.elementNameTable;
            attributeNameTable = parent.attributeNameTable;
            defaultNS = parent.defaultNS;
            declSeen = false;
        }

        /**
         * Makes associated state become collectible,
         * invalidating this context.
         * {@link #setParent} must be called before
         * this context may be used again.
         */
        void clear ()
        {
            parent = null;
            prefixTable = null;
            uriTable = null;
            elementNameTable = null;
            attributeNameTable = null;
            defaultNS = null;
        }


        /**
         * Declare a Namespace prefix for this context.
         *
         * @param prefix The prefix to declare.
         * @param uri The associated Namespace URI.
         * @see org.xml.sax.helpers.NamespaceSupport#declarePrefix
         */
        void declarePrefix (String prefix, String uri)
        {
                                // Lazy processing...
//          if (!declsOK)
//              throw new IllegalStateException (
//                  "can't declare any more prefixes in this context");
            if (!declSeen) {
                copyTables();
            }
            if (declarations == null) {
                declarations = new ArrayList<>();
            }

            prefix = prefix.intern();
            uri = uri.intern();
            if ("".equals(prefix)) {
                if ("".equals(uri)) {
                    defaultNS = null;
                } else {
                    defaultNS = uri;
                }
            } else {
```

```
                prefixTable.put(prefix, uri);
                uriTable.put(uri, prefix); // may wipe out another prefix
            }
            declarations.add(prefix);
        }


        /**
         * Process an XML qualified name in this context.
         *
         * @param qName The XML qualified name.
         * @param isAttribute true if this is an attribute name.
         * @return An array of three strings containing the
         *         URI part (or empty string), the local part,
         *         and the raw name, all internalized, or null
         *         if there is an undeclared prefix.
         * @see org.xml.sax.helpers.NamespaceSupport#processName
         */
        String [] processName (String qName, boolean isAttribute)
        {
            String name[];
            Map<String, String[]> table;

            // Select the appropriate table.
            if (isAttribute) {
                table = attributeNameTable;
            } else {
                table = elementNameTable;
            }

            // Start by looking in the cache, and
            // return immediately if the name
            // is already known in this content
            name = (String[])table.get(qName);
            if (name != null) {
                return name;
            }

            // We haven't seen this name in this
            // context before.  Maybe in the parent
            // context, but we can't assume prefix
            // bindings are the same.
            name = new String[3];
            name[2] = qName.intern();
            int index = qName.indexOf(':');


            // No prefix.
            if (index == -1) {
                if (isAttribute) {
                    if (qName == "xmlns" && namespaceDeclUris)
                        name[0] = NSDECL;
                    else
```

```
                    name[0] = "";
            } else if (defaultNS == null) {
                name[0] = "";
            } else {
                name[0] = defaultNS;
            }
            name[1] = name[2];
        }

        // Prefix
        else {
            String prefix = qName.substring(0, index);
            String local = qName.substring(index+1);
            String uri;
            if ("".equals(prefix)) {
                uri = defaultNS;
            } else {
                uri = (String)prefixTable.get(prefix);
            }
            if (uri == null
                    || (!isAttribute && "xmlns".equals (prefix))) {
                return null;
            }
            name[0] = uri;
            name[1] = local.intern();
        }

        // Save in the cache for future use.
        // (Could be shared with parent context...)
        table.put(name[2], name);
        return name;
    }


    /**
     * Look up the URI associated with a prefix in this context.
     *
     * @param prefix The prefix to look up.
     * @return The associated Namespace URI, or null if none is
     *         declared.
     * @see org.xml.sax.helpers.NamespaceSupport#getURI
     */
    String getURI (String prefix)
    {
        if ("".equals(prefix)) {
            return defaultNS;
        } else if (prefixTable == null) {
            return null;
        } else {
            return (String)prefixTable.get(prefix);
        }
    }

```

```java

        /**
         * Look up one of the prefixes associated with a URI in this context.
         *
         * <p>Since many prefixes may be mapped to the same URI,
         * the return value may be unreliable.</p>
         *
         * @param uri The URI to look up.
         * @return The associated prefix, or null if none is declared.
         * @see org.xml.sax.helpers.NamespaceSupport#getPrefix
         */
        String getPrefix (String uri)
        {
            if (uriTable == null) {
                return null;
            } else {
                return (String)uriTable.get(uri);
            }
        }


        /**
         * Return an enumeration of prefixes declared in this context.
         *
         * @return An enumeration of prefixes (possibly empty).
         * @see org.xml.sax.helpers.NamespaceSupport#getDeclaredPrefixes
         */
        Enumeration getDeclaredPrefixes ()
        {
            if (declarations == null) {
                return EMPTY_ENUMERATION;
            } else {
                return Collections.enumeration(declarations);
            }
        }

        /**
         * Return an enumeration of all prefixes currently in force.
         *
         * <p>The default prefix, if in force, is <em>not</em>
         * returned, and will have to be checked for separately.</p>
         *
         * @return An enumeration of prefixes (never empty).
         * @see org.xml.sax.helpers.NamespaceSupport#getPrefixes
         */
        Enumeration getPrefixes ()
        {
            if (prefixTable == null) {
                return EMPTY_ENUMERATION;
            } else {
                return Collections.enumeration(prefixTable.keySet());
            }
        }
```

```java
        ////////////////////////////////////////////////////////////////
        // Internal methods.
        ////////////////////////////////////////////////////////////////


        /**
         * Copy on write for the internal tables in this context.
         *
         * <p>This class is optimized for the normal case where most
         * elements do not contain Namespace declarations.</p>
         */
        private void copyTables ()
        {
            if (prefixTable != null) {
                prefixTable = new HashMap<>(prefixTable);
            } else {
                prefixTable = new HashMap<>();
            }
            if (uriTable != null) {
                uriTable = new HashMap<>(uriTable);
            } else {
                uriTable = new HashMap<>();
            }
            elementNameTable = new HashMap<>();
            attributeNameTable = new HashMap<>();
            declSeen = true;
        }



        ////////////////////////////////////////////////////////////////
        // Protected state.
        ////////////////////////////////////////////////////////////////

        Map<String, String> prefixTable;
        Map<String, String> uriTable;
        Map<String, String[]> elementNameTable;
        Map<String, String[]> attributeNameTable;
        String defaultNS = null;



        ////////////////////////////////////////////////////////////////
        // Internal state.
        ////////////////////////////////////////////////////////////////

        private List<String> declarations = null;
        private boolean declSeen = false;
        private Context parent = null;
    }
```

```
}

// end of NamespaceSupport.java
```

49.6 NewInstance.java

Secuencia 508: org/xml/sax/helpers/NewInstance.java

```
/*
 * Copyright (c) 2001, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// NewInstance.java - create a new instance of a class by name.
// http://www.saxproject.org
// Written by Edwin Goei, edwingo@apache.org
// and by David Brownell, dbrownell@users.sourceforge.net
// NO WARRANTY!  This class is in the Public Domain.
// $Id: NewInstance.java,v 1.2 2005/06/10 03:50:50 jeffsuttor Exp $

package org.xml.sax.helpers;

import java.lang.reflect.Method;
import java.lang.reflect.InvocationTargetException;

/**
 * Create a new instance of a class by name.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
```

```
 *
 * <p>This class contains a static method for creating an instance of a
 * class from an explicit class name.  It tries to use the thread's context
 * ClassLoader if possible and falls back to using
 * Class.forName(String).</p>
 *
 * <p>This code is designed to compile and run on JDK version 1.1 and later
 * including versions of Java 2.</p>
 *
 * @author Edwin Goei, David Brownell
 * @version 2.0.1 (sax2r2)
 */
class NewInstance {
    private static final String DEFAULT_PACKAGE = "com.sun.org.apache.xerces.internal";
    /**
     * Creates a new instance of the specified class name
     *
     * Package private so this code is not exposed at the API level.
     */
    static Object newInstance (ClassLoader classLoader, String className)
        throws ClassNotFoundException, IllegalAccessException,
            InstantiationException
    {
        // make sure we have access to restricted packages
        boolean internal = false;
        if (System.getSecurityManager() != null) {
            if (className != null && className.startsWith(DEFAULT_PACKAGE)) {
                internal = true;
            }
        }

        Class driverClass;
        if (classLoader == null || internal) {
            driverClass = Class.forName(className);
        } else {
            driverClass = classLoader.loadClass(className);
        }
        return driverClass.newInstance();
    }

}
```

49.7 ParserAdapter.java

Secuencia 509: org/xml/sax/helpers/ParserAdapter.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// ParserAdapter.java - adapt a SAX1 Parser to a SAX2 XMLReader.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: ParserAdapter.java,v 1.3 2004/11/03 22:53:09 jsuttor Exp $

package org.xml.sax.helpers;

import java.io.IOException;
import java.util.Enumeration;
import java.util.Vector;

import org.xml.sax.Parser;      // deprecated
import org.xml.sax.InputSource;
import org.xml.sax.Locator;
import org.xml.sax.AttributeList; // deprecated
import org.xml.sax.EntityResolver;
import org.xml.sax.DTDHandler;
import org.xml.sax.DocumentHandler; // deprecated
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import org.xml.sax.XMLReader;
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;


/**
 * Adapt a SAX1 Parser as a SAX2 XMLReader.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
```

```
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class wraps a SAX1 {@link org.xml.sax.Parser Parser}
 * and makes it act as a SAX2 {@link org.xml.sax.XMLReader XMLReader},
 * with feature, property, and Namespace support.  Note
 * that it is not possible to report {@link org.xml.sax.ContentHandler#skippedEntity
 * skippedEntity} events, since SAX1 does not make that information available.</p>
 *
 * <p>This adapter does not test for duplicate Namespace-qualified
 * attribute names.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @version 2.0.1 (sax2r2)
 * @see org.xml.sax.helpers.XMLReaderAdapter
 * @see org.xml.sax.XMLReader
 * @see org.xml.sax.Parser
 */
public class ParserAdapter implements XMLReader, DocumentHandler
{
    private static SecuritySupport ss = new SecuritySupport();

    ////////////////////////////////////////////////////////////////////
    // Constructors.
    ////////////////////////////////////////////////////////////////////


    /**
     * Construct a new parser adapter.
     *
     * <p>Use the "org.xml.sax.parser" property to locate the
     * embedded SAX1 driver.</p>
     *
     * @exception SAXException If the embedded driver
     *            cannot be instantiated or if the
     *            org.xml.sax.parser property is not specified.
     */
    public ParserAdapter ()
      throws SAXException
    {
        super();

        String driver = ss.getSystemProperty("org.xml.sax.parser");

        try {
            setup(ParserFactory.makeParser());
        } catch (ClassNotFoundException e1) {
            throw new
                SAXException("Cannot find SAX1 driver class " +
                             driver, e1);
        } catch (IllegalAccessException e2) {
```

```
            throw new
                SAXException("SAX1 driver class " +
                             driver +
                             " found but cannot be loaded", e2);
        } catch (InstantiationException e3) {
            throw new
                SAXException("SAX1 driver class " +
                             driver +
                             " loaded but cannot be instantiated", e3);
        } catch (ClassCastException e4) {
            throw new
                SAXException("SAX1 driver class " +
                             driver +
                             " does not implement org.xml.sax.Parser");
        } catch (NullPointerException e5) {
            throw new
                SAXException("System property org.xml.sax.parser not specified");
        }
    }


    /**
     * Construct a new parser adapter.
     *
     * <p>Note that the embedded parser cannot be changed once the
     * adapter is created; to embed a different parser, allocate
     * a new ParserAdapter.</p>
     *
     * @param parser The SAX1 parser to embed.
     * @exception java.lang.NullPointerException If the parser parameter
     *            is null.
     */
    public ParserAdapter (Parser parser)
    {
        super();
        setup(parser);
    }


    /**
     * Internal setup method.
     *
     * @param parser The embedded parser.
     * @exception java.lang.NullPointerException If the parser parameter
     *            is null.
     */
    private void setup (Parser parser)
    {
        if (parser == null) {
            throw new
                NullPointerException("Parser argument must not be null");
        }
        this.parser = parser;
```

```
        atts = new AttributesImpl();
        nsSupport = new NamespaceSupport();
        attAdapter = new AttributeListAdapter();
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.XMLReader.
    ////////////////////////////////////////////////////////////////////


    //
    // Internal constants for the sake of convenience.
    //
    private final static String FEATURES = "http://xml.org/sax/features/";
    private final static String NAMESPACES = FEATURES + "namespaces";
    private final static String NAMESPACE_PREFIXES = FEATURES + "namespace-prefixes";
    private final static String XMLNS_URIs = FEATURES + "xmlns-uris";


    /**
     * Set a feature flag for the parser.
     *
     * <p>The only features recognized are namespaces and
     * namespace-prefixes.</p>
     *
     * @param name The feature name, as a complete URI.
     * @param value The requested feature value.
     * @exception SAXNotRecognizedException If the feature
     *            can't be assigned or retrieved.
     * @exception SAXNotSupportedException If the feature
     *            can't be assigned that value.
     * @see org.xml.sax.XMLReader#setFeature
     */
    public void setFeature (String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (name.equals(NAMESPACES)) {
            checkNotParsing("feature", name);
            namespaces = value;
            if (!namespaces && !prefixes) {
                prefixes = true;
            }
        } else if (name.equals(NAMESPACE_PREFIXES)) {
            checkNotParsing("feature", name);
            prefixes = value;
            if (!prefixes && !namespaces) {
                namespaces = true;
            }
        } else if (name.equals(XMLNS_URIs)) {
            checkNotParsing("feature", name);
            uris = value;
```

```
        } else {
            throw new SAXNotRecognizedException("Feature: " + name);
        }
    }


    /**
     * Check a parser feature flag.
     *
     * <p>The only features recognized are namespaces and
     * namespace-prefixes.</p>
     *
     * @param name The feature name, as a complete URI.
     * @return The current feature value.
     * @exception SAXNotRecognizedException If the feature
     *            value can't be assigned or retrieved.
     * @exception SAXNotSupportedException If the
     *            feature is not currently readable.
     * @see org.xml.sax.XMLReader#setFeature
     */
    public boolean getFeature (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (name.equals(NAMESPACES)) {
            return namespaces;
        } else if (name.equals(NAMESPACE_PREFIXES)) {
            return prefixes;
        } else if (name.equals(XMLNS_URIs)) {
            return uris;
        } else {
            throw new SAXNotRecognizedException("Feature: " + name);
        }
    }


    /**
     * Set a parser property.
     *
     * <p>No properties are currently recognized.</p>
     *
     * @param name The property name.
     * @param value The property value.
     * @exception SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved.
     * @exception SAXNotSupportedException If the property
     *            can't be assigned that value.
     * @see org.xml.sax.XMLReader#setProperty
     */
    public void setProperty (String name, Object value)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        throw new SAXNotRecognizedException("Property: " + name);
    }
```

```java


    /**
     * Get a parser property.
     *
     * <p>No properties are currently recognized.</p>
     *
     * @param name The property name.
     * @return The property value.
     * @exception SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved.
     * @exception SAXNotSupportedException If the property
     *            value is not currently readable.
     * @see org.xml.sax.XMLReader#getProperty
     */
    public Object getProperty (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        throw new SAXNotRecognizedException("Property: " + name);
    }


    /**
     * Set the entity resolver.
     *
     * @param resolver The new entity resolver.
     * @see org.xml.sax.XMLReader#setEntityResolver
     */
    public void setEntityResolver (EntityResolver resolver)
    {
        entityResolver = resolver;
    }


    /**
     * Return the current entity resolver.
     *
     * @return The current entity resolver, or null if none was supplied.
     * @see org.xml.sax.XMLReader#getEntityResolver
     */
    public EntityResolver getEntityResolver ()
    {
        return entityResolver;
    }


    /**
     * Set the DTD handler.
     *
     * @param handler the new DTD handler
     * @see org.xml.sax.XMLReader#setEntityResolver
     */
    public void setDTDHandler (DTDHandler handler)
```

```
    {
        dtdHandler = handler;
    }


    /**
     * Return the current DTD handler.
     *
     * @return the current DTD handler, or null if none was supplied
     * @see org.xml.sax.XMLReader#getEntityResolver
     */
    public DTDHandler getDTDHandler ()
    {
        return dtdHandler;
    }


    /**
     * Set the content handler.
     *
     * @param handler the new content handler
     * @see org.xml.sax.XMLReader#setEntityResolver
     */
    public void setContentHandler (ContentHandler handler)
    {
        contentHandler = handler;
    }


    /**
     * Return the current content handler.
     *
     * @return The current content handler, or null if none was supplied.
     * @see org.xml.sax.XMLReader#getEntityResolver
     */
    public ContentHandler getContentHandler ()
    {
        return contentHandler;
    }


    /**
     * Set the error handler.
     *
     * @param handler The new error handler.
     * @see org.xml.sax.XMLReader#setEntityResolver
     */
    public void setErrorHandler (ErrorHandler handler)
    {
        errorHandler = handler;
    }


```

```
    /**
     * Return the current error handler.
     *
     * @return The current error handler, or null if none was supplied.
     * @see org.xml.sax.XMLReader#getEntityResolver
     */
    public ErrorHandler getErrorHandler ()
    {
        return errorHandler;
    }


    /**
     * Parse an XML document.
     *
     * @param systemId The absolute URL of the document.
     * @exception java.io.IOException If there is a problem reading
     *            the raw content of the document.
     * @exception SAXException If there is a problem
     *            processing the document.
     * @see #parse(org.xml.sax.InputSource)
     * @see org.xml.sax.Parser#parse(java.lang.String)
     */
    public void parse (String systemId)
        throws IOException, SAXException
    {
        parse(new InputSource(systemId));
    }


    /**
     * Parse an XML document.
     *
     * @param input An input source for the document.
     * @exception java.io.IOException If there is a problem reading
     *            the raw content of the document.
     * @exception SAXException If there is a problem
     *            processing the document.
     * @see #parse(java.lang.String)
     * @see org.xml.sax.Parser#parse(org.xml.sax.InputSource)
     */
    public void parse (InputSource input)
        throws IOException, SAXException
    {
        if (parsing) {
            throw new SAXException("Parser is already in use");
        }
        setupParser();
        parsing = true;
        try {
            parser.parse(input);
        } finally {
            parsing = false;
```

```
        }
        parsing = false;
    }



    ////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.DocumentHandler.
    ////////////////////////////////////////////////////////////////


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 document locator event.
     *
     * @param locator A document locator.
     * @see org.xml.sax.ContentHandler#setDocumentLocator
     */
    public void setDocumentLocator (Locator locator)
    {
        this.locator = locator;
        if (contentHandler != null) {
            contentHandler.setDocumentLocator(locator);
        }
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 start document event.
     *
     * @exception SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.DocumentHandler#startDocument
     */
    public void startDocument ()
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.startDocument();
        }
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 end document event.
     *
     * @exception SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.DocumentHandler#endDocument
     */
    public void endDocument ()
```

```
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.endDocument();
        }
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 startElement event.
     *
     * <p>If necessary, perform Namespace processing.</p>
     *
     * @param qName The qualified (prefixed) name.
     * @param qAtts The XML attribute list (with qnames).
     * @exception SAXException The client may raise a
     *            processing exception.
     */
    public void startElement (String qName, AttributeList qAtts)
        throws SAXException
    {
                                // These are exceptions from the
                                // first pass; they should be
                                // ignored if there's a second pass,
                                // but reported otherwise.
        Vector exceptions = null;

                                // If we're not doing Namespace
                                // processing, dispatch this quickly.
        if (!namespaces) {
            if (contentHandler != null) {
                attAdapter.setAttributeList(qAtts);
                contentHandler.startElement("", "", qName.intern(),
                                            attAdapter);
            }
            return;
        }


                                // OK, we're doing Namespace processing.
        nsSupport.pushContext();
        int length = qAtts.getLength();

                                // First pass:  handle NS decls
        for (int i = 0; i < length; i++) {
            String attQName = qAtts.getName(i);

            if (!attQName.startsWith("xmlns"))
                continue;
                                // Could be a declaration...
            String prefix;
            int n = attQName.indexOf(':');
```

```
                            // xmlns=...
            if (n == -1 && attQName.length () == 5) {
                prefix = "";
            } else if (n != 5) {
                // XML namespaces spec doesn't discuss "xmlnsf:oo"
                // (and similarly named) attributes ... at most, warn
                continue;
            } else              // xmlns:foo=...
                prefix = attQName.substring(n+1);

            String value = qAtts.getValue(i);
            if (!nsSupport.declarePrefix(prefix, value)) {
                reportError("Illegal Namespace prefix: " + prefix);
                continue;
            }
            if (contentHandler != null)
                contentHandler.startPrefixMapping(prefix, value);
        }

                            // Second pass: copy all relevant
                            // attributes into the SAX2 AttributeList
                            // using updated prefix bindings
        atts.clear();
        for (int i = 0; i < length; i++) {
            String attQName = qAtts.getName(i);
            String type = qAtts.getType(i);
            String value = qAtts.getValue(i);

                            // Declaration?
            if (attQName.startsWith("xmlns")) {
                String prefix;
                int n = attQName.indexOf(':');

                if (n == -1 && attQName.length () == 5) {
                    prefix = "";
                } else if (n != 5) {
                    // XML namespaces spec doesn't discuss "xmlnsf:oo"
                    // (and similarly named) attributes ... ignore
                    prefix = null;
                } else {
                    prefix = attQName.substring(6);
                }
                            // Yes, decl:  report or prune
                if (prefix != null) {
                    if (prefixes) {
                        if (uris)
                            // note funky case:  localname can be null
                            // when declaring the default prefix, and
                            // yet the uri isn't null.
                            atts.addAttribute (nsSupport.XMLNS, prefix,
                                    attQName.intern(), type, value);
                        else
```

```
                        atts.addAttribute ("", "",
                                attQName.intern(), type, value);
                }
                continue;
            }
        }

                            // Not a declaration -- report
        try {
            String attName[] = processName(attQName, true, true);
            atts.addAttribute(attName[0], attName[1], attName[2],
                              type, value);
        } catch (SAXException e) {
            if (exceptions == null)
                exceptions = new Vector();
            exceptions.addElement(e);
            atts.addAttribute("", attQName, attQName, type, value);
        }
    }

    // now handle the deferred exception reports
    if (exceptions != null && errorHandler != null) {
        for (int i = 0; i < exceptions.size(); i++)
            errorHandler.error((SAXParseException)
                            (exceptions.elementAt(i)));
    }

                            // OK, finally report the event.
    if (contentHandler != null) {
        String name[] = processName(qName, false, false);
        contentHandler.startElement(name[0], name[1], name[2], atts);
    }
}


/**
 * Adapter implementation method; do not call.
 * Adapt a SAX1 end element event.
 *
 * @param qName The qualified (prefixed) name.
 * @exception SAXException The client may raise a
 *            processing exception.
 * @see org.xml.sax.DocumentHandler#endElement
 */
public void endElement (String qName)
    throws SAXException
{
                            // If we're not doing Namespace
                            // processing, dispatch this quickly.
    if (!namespaces) {
        if (contentHandler != null) {
            contentHandler.endElement("", "", qName.intern());
        }
```

```
            return;
        }

                                // Split the name.
        String names[] = processName(qName, false, false);
        if (contentHandler != null) {
            contentHandler.endElement(names[0], names[1], names[2]);
            Enumeration prefixes = nsSupport.getDeclaredPrefixes();
            while (prefixes.hasMoreElements()) {
                String prefix = (String)prefixes.nextElement();
                contentHandler.endPrefixMapping(prefix);
            }
        }
        nsSupport.popContext();
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 characters event.
     *
     * @param ch An array of characters.
     * @param start The starting position in the array.
     * @param length The number of characters to use.
     * @exception SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.DocumentHandler#characters
     */
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.characters(ch, start, length);
        }
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 ignorable whitespace event.
     *
     * @param ch An array of characters.
     * @param start The starting position in the array.
     * @param length The number of characters to use.
     * @exception SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.DocumentHandler#ignorableWhitespace
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.ignorableWhitespace(ch, start, length);
```

```
        }
    }


    /**
     * Adapter implementation method; do not call.
     * Adapt a SAX1 processing instruction event.
     *
     * @param target The processing instruction target.
     * @param data The remainder of the processing instruction
     * @exception SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.DocumentHandler#processingInstruction
     */
    public void processingInstruction (String target, String data)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.processingInstruction(target, data);
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Internal utility methods.
    ////////////////////////////////////////////////////////////////////


    /**
     * Initialize the parser before each run.
     */
    private void setupParser ()
    {
        // catch an illegal "nonsense" state.
        if (!prefixes && !namespaces)
            throw new IllegalStateException ();

        nsSupport.reset();
        if (uris)
            nsSupport.setNamespaceDeclUris (true);

        if (entityResolver != null) {
            parser.setEntityResolver(entityResolver);
        }
        if (dtdHandler != null) {
            parser.setDTDHandler(dtdHandler);
        }
        if (errorHandler != null) {
            parser.setErrorHandler(errorHandler);
        }
        parser.setDocumentHandler(this);
        locator = null;
```

```
    }


    /**
     * Process a qualified (prefixed) name.
     *
     * <p>If the name has an undeclared prefix, use only the qname
     * and make an ErrorHandler.error callback in case the app is
     * interested.</p>
     *
     * @param qName The qualified (prefixed) name.
     * @param isAttribute true if this is an attribute name.
     * @return The name split into three parts.
     * @exception SAXException The client may throw
     *            an exception if there is an error callback.
     */
    private String [] processName (String qName, boolean isAttribute,
                                   boolean useException)
        throws SAXException
    {
        String parts[] = nsSupport.processName(qName, nameParts,
                                               isAttribute);
        if (parts == null) {
            if (useException)
                throw makeException("Undeclared prefix: " + qName);
            reportError("Undeclared prefix: " + qName);
            parts = new String[3];
            parts[0] = parts[1] = "";
            parts[2] = qName.intern();
        }
        return parts;
    }


    /**
     * Report a non-fatal error.
     *
     * @param message The error message.
     * @exception SAXException The client may throw
     *            an exception.
     */
    void reportError (String message)
        throws SAXException
    {
        if (errorHandler != null)
            errorHandler.error(makeException(message));
    }


    /**
     * Construct an exception for the current context.
     *
     * @param message The error message.
```

```
     */
    private SAXParseException makeException (String message)
    {
        if (locator != null) {
            return new SAXParseException(message, locator);
        } else {
            return new SAXParseException(message, null, null, -1, -1);
        }
    }


    /**
     * Throw an exception if we are parsing.
     *
     * <p>Use this method to detect illegal feature or
     * property changes.</p>
     *
     * @param type The type of thing (feature or property).
     * @param name The feature or property name.
     * @exception SAXNotSupportedException If a
     *            document is currently being parsed.
     */
    private void checkNotParsing (String type, String name)
        throws SAXNotSupportedException
    {
        if (parsing) {
            throw new SAXNotSupportedException("Cannot change " +
                                               type + ' ' +
                                               name + " while parsing");

        }
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    private NamespaceSupport nsSupport;
    private AttributeListAdapter attAdapter;

    private boolean parsing = false;
    private String nameParts[] = new String[3];

    private Parser parser = null;

    private AttributesImpl atts = null;

                                // Features
    private boolean namespaces = true;
    private boolean prefixes = false;
    private boolean uris = false;
```

```
                                // Properties

                                // Handlers
    Locator locator;

    EntityResolver entityResolver = null;
    DTDHandler dtdHandler = null;
    ContentHandler contentHandler = null;
    ErrorHandler errorHandler = null;



    ////////////////////////////////////////////////////////////////////
    // Inner class to wrap an AttributeList when not doing NS proc.
    ////////////////////////////////////////////////////////////////////


    /**
     * Adapt a SAX1 AttributeList as a SAX2 Attributes object.
     *
     * <p>This class is in the Public Domain, and comes with NO
     * WARRANTY of any kind.</p>
     *
     * <p>This wrapper class is used only when Namespace support
     * is disabled -- it provides pretty much a direct mapping
     * from SAX1 to SAX2, except that names and types are
     * interned whenever requested.</p>
     */
    final class AttributeListAdapter implements Attributes
    {

        /**
         * Construct a new adapter.
         */
        AttributeListAdapter ()
        {
        }


        /**
         * Set the embedded AttributeList.
         *
         * <p>This method must be invoked before any of the others
         * can be used.</p>
         *
         * @param The SAX1 attribute list (with qnames).
         */
        void setAttributeList (AttributeList qAtts)
        {
            this.qAtts = qAtts;
        }

```

```java

        /**
         * Return the length of the attribute list.
         *
         * @return The number of attributes in the list.
         * @see org.xml.sax.Attributes#getLength
         */
        public int getLength ()
        {
            return qAtts.getLength();
        }


        /**
         * Return the Namespace URI of the specified attribute.
         *
         * @param The attribute's index.
         * @return Always the empty string.
         * @see org.xml.sax.Attributes#getURI
         */
        public String getURI (int i)
        {
            return "";
        }


        /**
         * Return the local name of the specified attribute.
         *
         * @param The attribute's index.
         * @return Always the empty string.
         * @see org.xml.sax.Attributes#getLocalName
         */
        public String getLocalName (int i)
        {
            return "";
        }


        /**
         * Return the qualified (prefixed) name of the specified attribute.
         *
         * @param The attribute's index.
         * @return The attribute's qualified name, internalized.
         */
        public String getQName (int i)
        {
            return qAtts.getName(i).intern();
        }


        /**
         * Return the type of the specified attribute.
```

```
         *
         * @param The attribute's index.
         * @return The attribute's type as an internalized string.
         */
        public String getType (int i)
        {
            return qAtts.getType(i).intern();
        }


        /**
         * Return the value of the specified attribute.
         *
         * @param The attribute's index.
         * @return The attribute's value.
         */
        public String getValue (int i)
        {
            return qAtts.getValue(i);
        }


        /**
         * Look up an attribute index by Namespace name.
         *
         * @param uri The Namespace URI or the empty string.
         * @param localName The local name.
         * @return The attributes index, or -1 if none was found.
         * @see org.xml.sax.Attributes#getIndex(java.lang.String,java.lang.String)
         */
        public int getIndex (String uri, String localName)
        {
            return -1;
        }


        /**
         * Look up an attribute index by qualified (prefixed) name.
         *
         * @param qName The qualified name.
         * @return The attributes index, or -1 if none was found.
         * @see org.xml.sax.Attributes#getIndex(java.lang.String)
         */
        public int getIndex (String qName)
        {
            int max = atts.getLength();
            for (int i = 0; i < max; i++) {
                if (qAtts.getName(i).equals(qName)) {
                    return i;
                }
            }
            return -1;
        }
```

```


        /**
         * Look up the type of an attribute by Namespace name.
         *
         * @param uri The Namespace URI
         * @param localName The local name.
         * @return The attribute's type as an internalized string.
         */
        public String getType (String uri, String localName)
        {
            return null;
        }


        /**
         * Look up the type of an attribute by qualified (prefixed) name.
         *
         * @param qName The qualified name.
         * @return The attribute's type as an internalized string.
         */
        public String getType (String qName)
        {
            return qAtts.getType(qName).intern();
        }


        /**
         * Look up the value of an attribute by Namespace name.
         *
         * @param uri The Namespace URI
         * @param localName The local name.
         * @return The attribute's value.
         */
        public String getValue (String uri, String localName)
        {
            return null;
        }


        /**
         * Look up the value of an attribute by qualified (prefixed) name.
         *
         * @param qName The qualified name.
         * @return The attribute's value.
         */
        public String getValue (String qName)
        {
            return qAtts.getValue(qName);
        }

        private AttributeList qAtts;
    }
```

```
}

// end of ParserAdapter.java
```

49.8 ParserFactory.java

Secuencia 510: org/xml/sax/helpers/ParserFactory.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// SAX parser factory.
// http://www.saxproject.org
// No warranty; no copyright -- use this as you will.
// $Id: ParserFactory.java,v 1.2 2004/11/03 22:53:09 jsuttor Exp $

package org.xml.sax.helpers;

import org.xml.sax.Parser;


/**
 * Java-specific class for dynamically loading SAX parsers.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p><strong>Note:</strong> This class is designed to work with the now-deprecated
```

```
 * SAX1 {@link org.xml.sax.Parser Parser} class.  SAX2 applications should use
 * {@link org.xml.sax.helpers.XMLReaderFactory XMLReaderFactory} instead.</p>
 *
 * <p>ParserFactory is not part of the platform-independent definition
 * of SAX; it is an additional convenience class designed
 * specifically for Java XML application writers.  SAX applications
 * can use the static methods in this class to allocate a SAX parser
 * dynamically at run-time based either on the value of the
 * `org.xml.sax.parser' system property or on a string containing the class
 * name.</p>
 *
 * <p>Note that the application still requires an XML parser that
 * implements SAX1.</p>
 *
 * @deprecated This class works with the deprecated
 *             {@link org.xml.sax.Parser Parser}
 *             interface.
 * @since SAX 1.0
 * @author David Megginson
 * @version 2.0.1 (sax2r2)
 */
public class ParserFactory {
    private static SecuritySupport ss = new SecuritySupport();

    /**
     * Private null constructor.
     */
    private ParserFactory ()
    {
    }


    /**
     * Create a new SAX parser using the `org.xml.sax.parser' system property.
     *
     * <p>The named class must exist and must implement the
     * {@link org.xml.sax.Parser Parser} interface.</p>
     *
     * @exception java.lang.NullPointerException There is no value
     *            for the `org.xml.sax.parser' system property.
     * @exception java.lang.ClassNotFoundException The SAX parser
     *            class was not found (check your CLASSPATH).
     * @exception IllegalAccessException The SAX parser class was
     *            found, but you do not have permission to load
     *            it.
     * @exception InstantiationException The SAX parser class was
     *            found but could not be instantiated.
     * @exception java.lang.ClassCastException The SAX parser class
     *            was found and instantiated, but does not implement
     *            org.xml.sax.Parser.
     * @see #makeParser(java.lang.String)
     * @see org.xml.sax.Parser
     */
```

```
    public static Parser makeParser ()
        throws ClassNotFoundException,
        IllegalAccessException,
        InstantiationException,
        NullPointerException,
        ClassCastException
    {
        String className = ss.getSystemProperty("org.xml.sax.parser");
        if (className == null) {
            throw new NullPointerException("No value for sax.parser property");
        } else {
            return makeParser(className);
        }
    }


    /**
     * Create a new SAX parser object using the class name provided.
     *
     * <p>The named class must exist and must implement the
     * {@link org.xml.sax.Parser Parser} interface.</p>
     *
     * @param className A string containing the name of the
     *                  SAX parser class.
     * @exception java.lang.ClassNotFoundException The SAX parser
     *            class was not found (check your CLASSPATH).
     * @exception IllegalAccessException The SAX parser class was
     *            found, but you do not have permission to load
     *            it.
     * @exception InstantiationException The SAX parser class was
     *            found but could not be instantiated.
     * @exception java.lang.ClassCastException The SAX parser class
     *            was found and instantiated, but does not implement
     *            org.xml.sax.Parser.
     * @see #makeParser()
     * @see org.xml.sax.Parser
     */
    public static Parser makeParser (String className)
        throws ClassNotFoundException,
        IllegalAccessException,
        InstantiationException,
        ClassCastException
    {
        return (Parser) NewInstance.newInstance (
                ss.getContextClassLoader(), className);
    }

}
```

49.9 SecuritySupport.java

Secuencia 511: org/xml/sax/helpers/SecuritySupport.java

```
/*
```

```
 * Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

package org.xml.sax.helpers;

import java.io.*;
import java.security.*;

/**
 * This class is duplicated for each JAXP subpackage so keep it in sync.
 * It is package private and therefore is not exposed as part of the JAXP
 * API.
 *
 * Security related methods that only work on J2SE 1.2 and newer.
 */
class SecuritySupport  {


    ClassLoader getContextClassLoader() throws SecurityException{
        return (ClassLoader)
                AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                ClassLoader cl = null;
                //try {
                cl = Thread.currentThread().getContextClassLoader();
                //} catch (SecurityException ex) { }

                if (cl == null)
                    cl = ClassLoader.getSystemClassLoader();

                return cl;
            }
```

```
        });
    }

    String getSystemProperty(final String propName) {
        return (String)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return System.getProperty(propName);
                }
            });
    }

    FileInputStream getFileInputStream(final File file)
        throws FileNotFoundException
    {
        try {
            return (FileInputStream)
                AccessController.doPrivileged(new PrivilegedExceptionAction() {
                    public Object run() throws FileNotFoundException {
                        return new FileInputStream(file);
                    }
                });
        } catch (PrivilegedActionException e) {
            throw (FileNotFoundException)e.getException();
        }
    }

    InputStream getResourceAsStream(final ClassLoader cl,
                                    final String name)
    {
        return (InputStream)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    InputStream ris;
                    if (cl == null) {
                        ris = Object.class.getResourceAsStream(name);
                    } else {
                        ris = cl.getResourceAsStream(name);
                    }
                    return ris;
                }
            });
    }

    boolean doesFileExist(final File f) {
    return ((Boolean)
            AccessController.doPrivileged(new PrivilegedAction() {
                public Object run() {
                    return new Boolean(f.exists());
                }
            })).booleanValue();
    }

```

```
}
```

49.10 XMLFilterImpl.java

Secuencia 512: org/xml/sax/helpers/XMLFilterImpl.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// XMLFilterImpl.java - base SAX2 filter implementation.
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the Public Domain.
// $Id: XMLFilterImpl.java,v 1.3 2004/11/03 22:53:09 jsuttor Exp $

package org.xml.sax.helpers;

import java.io.IOException;

import org.xml.sax.XMLReader;
import org.xml.sax.XMLFilter;
import org.xml.sax.InputSource;
import org.xml.sax.Locator;
import org.xml.sax.Attributes;
import org.xml.sax.EntityResolver;
import org.xml.sax.DTDHandler;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.SAXNotSupportedException;
import org.xml.sax.SAXNotRecognizedException;
```

```


/**
 * Base class for deriving an XML filter.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class is designed to sit between an {@link org.xml.sax.XMLReader
 * XMLReader} and the client application's event handlers.  By default, it
 * does nothing but pass requests up to the reader and events
 * on to the handlers unmodified, but subclasses can override
 * specific methods to modify the event stream or the configuration
 * requests as they pass through.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.XMLFilter
 * @see org.xml.sax.XMLReader
 * @see org.xml.sax.EntityResolver
 * @see org.xml.sax.DTDHandler
 * @see org.xml.sax.ContentHandler
 * @see org.xml.sax.ErrorHandler
 */
public class XMLFilterImpl
    implements XMLFilter, EntityResolver, DTDHandler, ContentHandler, ErrorHandler
{


    ////////////////////////////////////////////////////////////////////
    // Constructors.
    ////////////////////////////////////////////////////////////////////


    /**
     * Construct an empty XML filter, with no parent.
     *
     * <p>This filter will have no parent: you must assign a parent
     * before you start a parse or do any configuration with
     * setFeature or setProperty, unless you use this as a pure event
     * consumer rather than as an {@link XMLReader}.</p>
     *
     * @see org.xml.sax.XMLReader#setFeature
     * @see org.xml.sax.XMLReader#setProperty
     * @see #setParent
     */
    public XMLFilterImpl ()
    {
        super();
```

```
    }


    /**
     * Construct an XML filter with the specified parent.
     *
     * @see #setParent
     * @see #getParent
     */
    public XMLFilterImpl (XMLReader parent)
    {
        super();
        setParent(parent);
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.XMLFilter.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set the parent reader.
     *
     * <p>This is the {@link org.xml.sax.XMLReader XMLReader} from which
     * this filter will obtain its events and to which it will pass its
     * configuration requests.  The parent may itself be another filter.</p>
     *
     * <p>If there is no parent reader set, any attempt to parse
     * or to set or get a feature or property will fail.</p>
     *
     * @param parent The parent XML reader.
     * @see #getParent
     */
    public void setParent (XMLReader parent)
    {
        this.parent = parent;
    }


    /**
     * Get the parent reader.
     *
     * @return The parent XML reader, or null if none is set.
     * @see #setParent
     */
    public XMLReader getParent ()
    {
        return parent;
    }


```

```
    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.XMLReader.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set the value of a feature.
     *
     * <p>This will always fail if the parent is null.</p>
     *
     * @param name The feature name.
     * @param value The requested feature value.
     * @exception org.xml.sax.SAXNotRecognizedException If the feature
     *            value can't be assigned or retrieved from the parent.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            parent recognizes the feature name but
     *            cannot set the requested value.
     */
    public void setFeature (String name, boolean value)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (parent != null) {
            parent.setFeature(name, value);
        } else {
            throw new SAXNotRecognizedException("Feature: " + name);
        }
    }


    /**
     * Look up the value of a feature.
     *
     * <p>This will always fail if the parent is null.</p>
     *
     * @param name The feature name.
     * @return The current value of the feature.
     * @exception org.xml.sax.SAXNotRecognizedException If the feature
     *            value can't be assigned or retrieved from the parent.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            parent recognizes the feature name but
     *            cannot determine its value at this time.
     */
    public boolean getFeature (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (parent != null) {
            return parent.getFeature(name);
        } else {
            throw new SAXNotRecognizedException("Feature: " + name);
        }
    }
```

```java

    /**
     * Set the value of a property.
     *
     * <p>This will always fail if the parent is null.</p>
     *
     * @param name The property name.
     * @param value The requested property value.
     * @exception org.xml.sax.SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved from the parent.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            parent recognizes the property name but
     *            cannot set the requested value.
     */
    public void setProperty (String name, Object value)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (parent != null) {
            parent.setProperty(name, value);
        } else {
            throw new SAXNotRecognizedException("Property: " + name);
        }
    }


    /**
     * Look up the value of a property.
     *
     * @param name The property name.
     * @return The current value of the property.
     * @exception org.xml.sax.SAXNotRecognizedException If the property
     *            value can't be assigned or retrieved from the parent.
     * @exception org.xml.sax.SAXNotSupportedException When the
     *            parent recognizes the property name but
     *            cannot determine its value at this time.
     */
    public Object getProperty (String name)
        throws SAXNotRecognizedException, SAXNotSupportedException
    {
        if (parent != null) {
            return parent.getProperty(name);
        } else {
            throw new SAXNotRecognizedException("Property: " + name);
        }
    }


    /**
     * Set the entity resolver.
     *
     * @param resolver The new entity resolver.
     */
    public void setEntityResolver (EntityResolver resolver)
```

```
    {
        entityResolver = resolver;
    }


    /**
     * Get the current entity resolver.
     *
     * @return The current entity resolver, or null if none was set.
     */
    public EntityResolver getEntityResolver ()
    {
        return entityResolver;
    }


    /**
     * Set the DTD event handler.
     *
     * @param handler the new DTD handler
     */
    public void setDTDHandler (DTDHandler handler)
    {
        dtdHandler = handler;
    }


    /**
     * Get the current DTD event handler.
     *
     * @return The current DTD handler, or null if none was set.
     */
    public DTDHandler getDTDHandler ()
    {
        return dtdHandler;
    }


    /**
     * Set the content event handler.
     *
     * @param handler the new content handler
     */
    public void setContentHandler (ContentHandler handler)
    {
        contentHandler = handler;
    }


    /**
     * Get the content event handler.
     *
     * @return The current content handler, or null if none was set.
```

```
     */
    public ContentHandler getContentHandler ()
    {
        return contentHandler;
    }


    /**
     * Set the error event handler.
     *
     * @param handler the new error handler
     */
    public void setErrorHandler (ErrorHandler handler)
    {
        errorHandler = handler;
    }


    /**
     * Get the current error event handler.
     *
     * @return The current error handler, or null if none was set.
     */
    public ErrorHandler getErrorHandler ()
    {
        return errorHandler;
    }


    /**
     * Parse a document.
     *
     * @param input The input source for the document entity.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     */
    public void parse (InputSource input)
        throws SAXException, IOException
    {
        setupParse();
        parent.parse(input);
    }


    /**
     * Parse a document.
     *
     * @param systemId The system identifier as a fully-qualified URI.
     * @exception org.xml.sax.SAXException Any SAX exception, possibly
     *            wrapping another exception.
```

```
     * @exception java.io.IOException An IO exception from the parser,
     *            possibly from a byte stream or character stream
     *            supplied by the application.
     */
    public void parse (String systemId)
        throws SAXException, IOException
    {
        parse(new InputSource(systemId));
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.EntityResolver.
    ////////////////////////////////////////////////////////////////////


    /**
     * Filter an external entity resolution.
     *
     * @param publicId The entity's public identifier, or null.
     * @param systemId The entity's system identifier.
     * @return A new InputSource or null for the default.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     * @exception java.io.IOException The client may throw an
     *            I/O-related exception while obtaining the
     *            new InputSource.
     */
    public InputSource resolveEntity (String publicId, String systemId)
        throws SAXException, IOException
    {
        if (entityResolver != null) {
            return entityResolver.resolveEntity(publicId, systemId);
        } else {
            return null;
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.DTDHandler.
    ////////////////////////////////////////////////////////////////////


    /**
     * Filter a notation declaration event.
     *
     * @param name The notation name.
     * @param publicId The notation's public identifier, or null.
     * @param systemId The notation's system identifier, or null.
     * @exception org.xml.sax.SAXException The client may throw
```

```
     *            an exception during processing.
     */
    public void notationDecl (String name, String publicId, String systemId)
        throws SAXException
    {
        if (dtdHandler != null) {
            dtdHandler.notationDecl(name, publicId, systemId);
        }
    }


    /**
     * Filter an unparsed entity declaration event.
     *
     * @param name The entity name.
     * @param publicId The entity's public identifier, or null.
     * @param systemId The entity's system identifier, or null.
     * @param notationName The name of the associated notation.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void unparsedEntityDecl (String name, String publicId,
                                    String systemId, String notationName)
        throws SAXException
    {
        if (dtdHandler != null) {
            dtdHandler.unparsedEntityDecl(name, publicId, systemId,
                                          notationName);
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.ContentHandler.
    ////////////////////////////////////////////////////////////////////


    /**
     * Filter a new document locator event.
     *
     * @param locator The document locator.
     */
    public void setDocumentLocator (Locator locator)
    {
        this.locator = locator;
        if (contentHandler != null) {
            contentHandler.setDocumentLocator(locator);
        }
    }


    /**
```

```
     * Filter a start document event.
     *
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void startDocument ()
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.startDocument();
        }
    }


    /**
     * Filter an end document event.
     *
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void endDocument ()
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.endDocument();
        }
    }


    /**
     * Filter a start Namespace prefix mapping event.
     *
     * @param prefix The Namespace prefix.
     * @param uri The Namespace URI.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.startPrefixMapping(prefix, uri);
        }
    }


    /**
     * Filter an end Namespace prefix mapping event.
     *
     * @param prefix The Namespace prefix.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
```

```
    public void endPrefixMapping (String prefix)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.endPrefixMapping(prefix);
        }
    }


    /**
     * Filter a start element event.
     *
     * @param uri The element's Namespace URI, or the empty string.
     * @param localName The element's local name, or the empty string.
     * @param qName The element's qualified (prefixed) name, or the empty
     *        string.
     * @param atts The element's attributes.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void startElement (String uri, String localName, String qName,
                              Attributes atts)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.startElement(uri, localName, qName, atts);
        }
    }


    /**
     * Filter an end element event.
     *
     * @param uri The element's Namespace URI, or the empty string.
     * @param localName The element's local name, or the empty string.
     * @param qName The element's qualified (prefixed) name, or the empty
     *        string.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void endElement (String uri, String localName, String qName)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.endElement(uri, localName, qName);
        }
    }


    /**
     * Filter a character data event.
     *
     * @param ch An array of characters.
```

```
     * @param start The starting position in the array.
     * @param length The number of characters to use from the array.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.characters(ch, start, length);
        }
    }


    /**
     * Filter an ignorable whitespace event.
     *
     * @param ch An array of characters.
     * @param start The starting position in the array.
     * @param length The number of characters to use from the array.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.ignorableWhitespace(ch, start, length);
        }
    }


    /**
     * Filter a processing instruction event.
     *
     * @param target The processing instruction target.
     * @param data The text following the target.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void processingInstruction (String target, String data)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.processingInstruction(target, data);
        }
    }


    /**
     * Filter a skipped entity event.
     *
     * @param name The name of the skipped entity.
```

```
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void skippedEntity (String name)
        throws SAXException
    {
        if (contentHandler != null) {
            contentHandler.skippedEntity(name);
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.ErrorHandler.
    ////////////////////////////////////////////////////////////////////


    /**
     * Filter a warning event.
     *
     * @param e The warning as an exception.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void warning (SAXParseException e)
        throws SAXException
    {
        if (errorHandler != null) {
            errorHandler.warning(e);
        }
    }


    /**
     * Filter an error event.
     *
     * @param e The error as an exception.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void error (SAXParseException e)
        throws SAXException
    {
        if (errorHandler != null) {
            errorHandler.error(e);
        }
    }


    /**
     * Filter a fatal error event.
     *
```

```
     * @param e The error as an exception.
     * @exception org.xml.sax.SAXException The client may throw
     *            an exception during processing.
     */
    public void fatalError (SAXParseException e)
        throws SAXException
    {
        if (errorHandler != null) {
            errorHandler.fatalError(e);
        }
    }



    ////////////////////////////////////////////////////////////////////
    // Internal methods.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set up before a parse.
     *
     * <p>Before every parse, check whether the parent is
     * non-null, and re-register the filter for all of the
     * events.</p>
     */
    private void setupParse ()
    {
        if (parent == null) {
            throw new NullPointerException("No parent for filter");
        }
        parent.setEntityResolver(this);
        parent.setDTDHandler(this);
        parent.setContentHandler(this);
        parent.setErrorHandler(this);
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    private XMLReader parent = null;
    private Locator locator = null;
    private EntityResolver entityResolver = null;
    private DTDHandler dtdHandler = null;
    private ContentHandler contentHandler = null;
    private ErrorHandler errorHandler = null;

}

// end of XMLFilterImpl.java
```

49.11 XMLReaderAdapter.java

Secuencia 513: org/xml/sax/helpers/XMLReaderAdapter.java

```
/*
 * Copyright (c) 2000, 2005, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// XMLReaderAdapter.java - adapt an SAX2 XMLReader to a SAX1 Parser
// http://www.saxproject.org
// Written by David Megginson
// NO WARRANTY!  This class is in the public domain.
// $Id: XMLReaderAdapter.java,v 1.3 2004/11/03 22:53:09 jsuttor Exp $

package org.xml.sax.helpers;

import java.io.IOException;
import java.util.Locale;

import org.xml.sax.Parser;      // deprecated
import org.xml.sax.Locator;
import org.xml.sax.InputSource;
import org.xml.sax.AttributeList; // deprecated
import org.xml.sax.EntityResolver;
import org.xml.sax.DTDHandler;
import org.xml.sax.DocumentHandler; // deprecated
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;

import org.xml.sax.XMLReader;
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.SAXNotSupportedException;
```

```


/**
 * Adapt a SAX2 XMLReader as a SAX1 Parser.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class wraps a SAX2 {@link org.xml.sax.XMLReader XMLReader}
 * and makes it act as a SAX1 {@link org.xml.sax.Parser Parser}.  The XMLReader
 * must support a true value for the
 * http://xml.org/sax/features/namespace-prefixes property or parsing will fail
 * with a {@link org.xml.sax.SAXException SAXException}; if the XMLReader
 * supports a false value for the http://xml.org/sax/features/namespaces
 * property, that will also be used to improve efficiency.</p>
 *
 * @since SAX 2.0
 * @author David Megginson
 * @see org.xml.sax.Parser
 * @see org.xml.sax.XMLReader
 */
public class XMLReaderAdapter implements Parser, ContentHandler
{


    ////////////////////////////////////////////////////////////////////
    // Constructor.
    ////////////////////////////////////////////////////////////////////


    /**
     * Create a new adapter.
     *
     * <p>Use the "org.xml.sax.driver" property to locate the SAX2
     * driver to embed.</p>
     *
     * @exception org.xml.sax.SAXException If the embedded driver
     *            cannot be instantiated or if the
     *            org.xml.sax.driver property is not specified.
     */
    public XMLReaderAdapter ()
      throws SAXException
    {
        setup(XMLReaderFactory.createXMLReader());
    }


    /**
     * Create a new adapter.
```

```
     *
     * <p>Create a new adapter, wrapped around a SAX2 XMLReader.
     * The adapter will make the XMLReader act like a SAX1
     * Parser.</p>
     *
     * @param xmlReader The SAX2 XMLReader to wrap.
     * @exception java.lang.NullPointerException If the argument is null.
     */
    public XMLReaderAdapter (XMLReader xmlReader)
    {
        setup(xmlReader);
    }



    /**
     * Internal setup.
     *
     * @param xmlReader The embedded XMLReader.
     */
    private void setup (XMLReader xmlReader)
    {
        if (xmlReader == null) {
            throw new NullPointerException("XMLReader must not be null");
        }
        this.xmlReader = xmlReader;
        qAtts = new AttributesAdapter();
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.Parser.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set the locale for error reporting.
     *
     * <p>This is not supported in SAX2, and will always throw
     * an exception.</p>
     *
     * @param locale the locale for error reporting.
     * @see org.xml.sax.Parser#setLocale
     * @exception org.xml.sax.SAXException Thrown unless overridden.
     */
    public void setLocale (Locale locale)
        throws SAXException
    {
        throw new SAXNotSupportedException("setLocale not supported");
    }


```

```
    /**
     * Register the entity resolver.
     *
     * @param resolver The new resolver.
     * @see org.xml.sax.Parser#setEntityResolver
     */
    public void setEntityResolver (EntityResolver resolver)
    {
        xmlReader.setEntityResolver(resolver);
    }


    /**
     * Register the DTD event handler.
     *
     * @param handler The new DTD event handler.
     * @see org.xml.sax.Parser#setDTDHandler
     */
    public void setDTDHandler (DTDHandler handler)
    {
        xmlReader.setDTDHandler(handler);
    }


    /**
     * Register the SAX1 document event handler.
     *
     * <p>Note that the SAX1 document handler has no Namespace
     * support.</p>
     *
     * @param handler The new SAX1 document event handler.
     * @see org.xml.sax.Parser#setDocumentHandler
     */
    public void setDocumentHandler (DocumentHandler handler)
    {
        documentHandler = handler;
    }


    /**
     * Register the error event handler.
     *
     * @param handler The new error event handler.
     * @see org.xml.sax.Parser#setErrorHandler
     */
    public void setErrorHandler (ErrorHandler handler)
    {
        xmlReader.setErrorHandler(handler);
    }


    /**
     * Parse the document.
```

```
     *
     * <p>This method will throw an exception if the embedded
     * XMLReader does not support the
     * http://xml.org/sax/features/namespace-prefixes property.</p>
     *
     * @param systemId The absolute URL of the document.
     * @exception java.io.IOException If there is a problem reading
     *            the raw content of the document.
     * @exception org.xml.sax.SAXException If there is a problem
     *            processing the document.
     * @see #parse(org.xml.sax.InputSource)
     * @see org.xml.sax.Parser#parse(java.lang.String)
     */
    public void parse (String systemId)
        throws IOException, SAXException
    {
        parse(new InputSource(systemId));
    }


    /**
     * Parse the document.
     *
     * <p>This method will throw an exception if the embedded
     * XMLReader does not support the
     * http://xml.org/sax/features/namespace-prefixes property.</p>
     *
     * @param input An input source for the document.
     * @exception java.io.IOException If there is a problem reading
     *            the raw content of the document.
     * @exception org.xml.sax.SAXException If there is a problem
     *            processing the document.
     * @see #parse(java.lang.String)
     * @see org.xml.sax.Parser#parse(org.xml.sax.InputSource)
     */
    public void parse (InputSource input)
        throws IOException, SAXException
    {
        setupXMLReader();
        xmlReader.parse(input);
    }


    /**
     * Set up the XML reader.
     */
    private void setupXMLReader ()
        throws SAXException
    {
        xmlReader.setFeature("http://xml.org/sax/features/namespace-prefixes", true);
        try {
            xmlReader.setFeature("http://xml.org/sax/features/namespaces",
                                 false);
```

```
        } catch (SAXException e) {
            // NO OP: it's just extra information, and we can ignore it
        }
        xmlReader.setContentHandler(this);
    }



    ////////////////////////////////////////////////////////////////////
    // Implementation of org.xml.sax.ContentHandler.
    ////////////////////////////////////////////////////////////////////


    /**
     * Set a document locator.
     *
     * @param locator The document locator.
     * @see org.xml.sax.ContentHandler#setDocumentLocator
     */
    public void setDocumentLocator (Locator locator)
    {
        if (documentHandler != null)
            documentHandler.setDocumentLocator(locator);
    }


    /**
     * Start document event.
     *
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#startDocument
     */
    public void startDocument ()
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.startDocument();
    }


    /**
     * End document event.
     *
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#endDocument
     */
    public void endDocument ()
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.endDocument();
```

```
    }


    /**
     * Adapt a SAX2 start prefix mapping event.
     *
     * @param prefix The prefix being mapped.
     * @param uri The Namespace URI being mapped to.
     * @see org.xml.sax.ContentHandler#startPrefixMapping
     */
    public void startPrefixMapping (String prefix, String uri)
    {
    }


    /**
     * Adapt a SAX2 end prefix mapping event.
     *
     * @param prefix The prefix being mapped.
     * @see org.xml.sax.ContentHandler#endPrefixMapping
     */
    public void endPrefixMapping (String prefix)
    {
    }


    /**
     * Adapt a SAX2 start element event.
     *
     * @param uri The Namespace URI.
     * @param localName The Namespace local name.
     * @param qName The qualified (prefixed) name.
     * @param atts The SAX2 attributes.
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#endDocument
     */
    public void startElement (String uri, String localName,
                              String qName, Attributes atts)
        throws SAXException
    {
        if (documentHandler != null) {
            qAtts.setAttributes(atts);
            documentHandler.startElement(qName, qAtts);
        }
    }


    /**
     * Adapt a SAX2 end element event.
     *
     * @param uri The Namespace URI.
     * @param localName The Namespace local name.
```

```
     * @param qName The qualified (prefixed) name.
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#endElement
     */
    public void endElement (String uri, String localName,
                            String qName)
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.endElement(qName);
    }


    /**
     * Adapt a SAX2 characters event.
     *
     * @param ch An array of characters.
     * @param start The starting position in the array.
     * @param length The number of characters to use.
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#characters
     */
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.characters(ch, start, length);
    }


    /**
     * Adapt a SAX2 ignorable whitespace event.
     *
     * @param ch An array of characters.
     * @param start The starting position in the array.
     * @param length The number of characters to use.
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#ignorableWhitespace
     */
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.ignorableWhitespace(ch, start, length);
    }


    /**
     * Adapt a SAX2 processing instruction event.
     *
```

```
     * @param target The processing instruction target.
     * @param data The remainder of the processing instruction
     * @exception org.xml.sax.SAXException The client may raise a
     *            processing exception.
     * @see org.xml.sax.ContentHandler#processingInstruction
     */
    public void processingInstruction (String target, String data)
        throws SAXException
    {
        if (documentHandler != null)
            documentHandler.processingInstruction(target, data);
    }


    /**
     * Adapt a SAX2 skipped entity event.
     *
     * @param name The name of the skipped entity.
     * @see org.xml.sax.ContentHandler#skippedEntity
     * @exception org.xml.sax.SAXException Throwable by subclasses.
     */
    public void skippedEntity (String name)
        throws SAXException
    {
    }



    ////////////////////////////////////////////////////////////////////
    // Internal state.
    ////////////////////////////////////////////////////////////////////

    XMLReader xmlReader;
    DocumentHandler documentHandler;
    AttributesAdapter qAtts;



    ////////////////////////////////////////////////////////////////////
    // Internal class.
    ////////////////////////////////////////////////////////////////////


    /**
     * Internal class to wrap a SAX2 Attributes object for SAX1.
     */
    final class AttributesAdapter implements AttributeList
    {
        AttributesAdapter ()
        {
        }


```

```
        /**
         * Set the embedded Attributes object.
         *
         * @param The embedded SAX2 Attributes.
         */
        void setAttributes (Attributes attributes)
        {
            this.attributes = attributes;
        }


        /**
         * Return the number of attributes.
         *
         * @return The length of the attribute list.
         * @see org.xml.sax.AttributeList#getLength
         */
        public int getLength ()
        {
            return attributes.getLength();
        }


        /**
         * Return the qualified (prefixed) name of an attribute by position.
         *
         * @return The qualified name.
         * @see org.xml.sax.AttributeList#getName
         */
        public String getName (int i)
        {
            return attributes.getQName(i);
        }


        /**
         * Return the type of an attribute by position.
         *
         * @return The type.
         * @see org.xml.sax.AttributeList#getType(int)
         */
        public String getType (int i)
        {
            return attributes.getType(i);
        }


        /**
         * Return the value of an attribute by position.
         *
         * @return The value.
         * @see org.xml.sax.AttributeList#getValue(int)
         */
```

```
        public String getValue (int i)
        {
            return attributes.getValue(i);
        }


        /**
         * Return the type of an attribute by qualified (prefixed) name.
         *
         * @return The type.
         * @see org.xml.sax.AttributeList#getType(java.lang.String)
         */
        public String getType (String qName)
        {
            return attributes.getType(qName);
        }


        /**
         * Return the value of an attribute by qualified (prefixed) name.
         *
         * @return The value.
         * @see org.xml.sax.AttributeList#getValue(java.lang.String)
         */
        public String getValue (String qName)
        {
            return attributes.getValue(qName);
        }

        private Attributes attributes;
    }

}

// end of XMLReaderAdapter.java
```

49.12 XMLReaderFactory.java

Secuencia 514: org/xml/sax/helpers/XMLReaderFactory.java

```
/*
 * Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

// XMLReaderFactory.java - factory for creating a new reader.
// http://www.saxproject.org
// Written by David Megginson
// and by David Brownell
// NO WARRANTY!  This class is in the Public Domain.
// $Id: XMLReaderFactory.java,v 1.2.2.1 2005/07/31 22:48:08 jeffsuttor Exp $

package org.xml.sax.helpers;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import org.xml.sax.XMLReader;
import org.xml.sax.SAXException;


/**
 * Factory for creating an XML reader.
 *
 * <blockquote>
 * <em>This module, both source code and documentation, is in the
 * Public Domain, and comes with <strong>NO WARRANTY</strong>.</em>
 * See <a href='http://www.saxproject.org'>http://www.saxproject.org</a>
 * for further information.
 * </blockquote>
 *
 * <p>This class contains static methods for creating an XML reader
 * from an explicit class name, or based on runtime defaults:</p>
 *
 * <pre>
 * try {
 *   XMLReader myReader = XMLReaderFactory.createXMLReader();
 * } catch (SAXException e) {
 *   System.err.println(e.getMessage());
 * }
 * </pre>
 *
 * <p><strong>Note to Distributions bundled with parsers:</strong>
 * You should modify the implementation of the no-arguments
 * <em>createXMLReader</em> to handle cases where the external
 * configuration mechanisms aren't set up.  That method should do its
 * best to return a parser when one is in the class path, even when
 * nothing bound its class name to <code>org.xml.sax.driver</code> so
```

```
 * those configuration mechanisms would see it.</p>
 *
 * @since SAX 2.0
 * @author David Megginson, David Brownell
 * @version 2.0.1 (sax2r2)
 */
final public class XMLReaderFactory
{
    /**
     * Private constructor.
     *
     * <p>This constructor prevents the class from being instantiated.</p>
     */
    private XMLReaderFactory ()
    {
    }

    private static final String property = "org.xml.sax.driver";
    private static SecuritySupport ss = new SecuritySupport();

    private static String _clsFromJar = null;
    private static boolean _jarread = false;
    /**
     * Attempt to create an XMLReader from system defaults.
     * In environments which can support it, the name of the XMLReader
     * class is determined by trying each these options in order, and
     * using the first one which succeeds:</p> <ul>
     *
     * <li>If the system property <code>org.xml.sax.driver</code>
     * has a value, that is used as an XMLReader class name. </li>
     *
     * <li>The JAR "Services API" is used to look for a class name
     * in the <em>META-INF/services/org.xml.sax.driver</em> file in
     * jarfiles available to the runtime.</li>
     *
     * <li> SAX parser distributions are strongly encouraged to provide
     * a default XMLReader class name that will take effect only when
     * previous options (on this list) are not successful.</li>
     *
     * <li>Finally, if {@link ParserFactory#makeParser()} can
     * return a system default SAX1 parser, that parser is wrapped in
     * a {@link ParserAdapter}.  (This is a migration aid for SAX1
     * environments, where the <code>org.xml.sax.parser</code> system
     * property will often be usable.) </li>
     *
     * </ul>
     *
     * <p> In environments such as small embedded systems, which can not
     * support that flexibility, other mechanisms to determine the default
     * may be used. </p>
     *
     * <p>Note that many Java environments allow system properties to be
     * initialized on a command line.  This means that <em>in most cases</em>
```

```
     * setting a good value for that property ensures that calls to this
     * method will succeed, except when security policies intervene.
     * This will also maximize application portability to older SAX
     * environments, with less robust implementations of this method.
     * </p>
     *
     * @return A new XMLReader.
     * @exception org.xml.sax.SAXException If no default XMLReader class
     *            can be identified and instantiated.
     * @see #createXMLReader(java.lang.String)
     */
    public static XMLReader createXMLReader ()
        throws SAXException
    {
        String          className = null;
        ClassLoader     cl = ss.getContextClassLoader();

        // 1. try the JVM-instance-wide system property
        try {
            className = ss.getSystemProperty(property);
        }
        catch (RuntimeException e) { /* continue searching */ }

        // 2. if that fails, try META-INF/services/
        if (className == null) {
            if (!_jarread) {
                _jarread = true;
                String      service = "META-INF/services/" + property;
                InputStream in;
                BufferedReader      reader;

                try {
                    if (cl != null) {
                        in = ss.getResourceAsStream(cl, service);

                        // If no provider found then try the current ClassLoader
                        if (in == null) {
                            cl = null;
                            in = ss.getResourceAsStream(cl, service);
                        }
                    } else {
                        // No Context ClassLoader, try the current ClassLoader
                        in = ss.getResourceAsStream(cl, service);
                    }

                    if (in != null) {
                        reader = new BufferedReader (new InputStreamReader (in, "UTF8"));
                        _clsFromJar = reader.readLine ();
                        in.close ();
                    }
                } catch (Exception e) {
                }
            }
```

```
            className = _clsFromJar;
        }

        // 3. Distro-specific fallback
        if (className == null) {
// BEGIN DISTRIBUTION-SPECIFIC

            // EXAMPLE:
            // className = "com.example.sax.XmlReader";
            // or a $JAVA_HOME/jre/lib/*properties setting...
            className = "com.sun.org.apache.xerces.internal.parsers.SAXParser";

// END DISTRIBUTION-SPECIFIC
        }

        // do we know the XMLReader implementation class yet?
        if (className != null)
            return loadClass (cl, className);

        // 4. panic -- adapt any SAX1 parser
        try {
            return new ParserAdapter (ParserFactory.makeParser ());
        } catch (Exception e) {
            throw new SAXException ("Can't create default XMLReader; "
                    + "is system property org.xml.sax.driver set?");
        }
    }


    /**
     * Attempt to create an XML reader from a class name.
     *
     * <p>Given a class name, this method attempts to load
     * and instantiate the class as an XML reader.</p>
     *
     * <p>Note that this method will not be usable in environments where
     * the caller (perhaps an applet) is not permitted to load classes
     * dynamically.</p>
     *
     * @return A new XML reader.
     * @exception org.xml.sax.SAXException If the class cannot be
     *            loaded, instantiated, and cast to XMLReader.
     * @see #createXMLReader()
     */
    public static XMLReader createXMLReader (String className)
        throws SAXException
    {
        return loadClass (ss.getContextClassLoader(), className);
    }

    private static XMLReader loadClass (ClassLoader loader, String className)
    throws SAXException
    {
```

```
        try {
            return (XMLReader) NewInstance.newInstance (loader, className);
        } catch (ClassNotFoundException e1) {
            throw new SAXException("SAX2 driver class " + className +
                                   " not found", e1);
        } catch (IllegalAccessException e2) {
            throw new SAXException("SAX2 driver class " + className +
                                   " found but cannot be loaded", e2);
        } catch (InstantiationException e3) {
            throw new SAXException("SAX2 driver class " + className +
           " loaded but cannot be instantiated (no empty public constructor?)",
                                   e3);
        } catch (ClassCastException e4) {
            throw new SAXException("SAX2 driver class " + className +
                                   " does not implement XMLReader", e4);
        }
    }
}
```